НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему мобільний додаток системи управління дипломними проектами.

Виконав: студент ___4_ курсу, групи_ТВ-51_

____Денисюк Віталій Вікторович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник _к.т.н., доц. Коваль Олександр Васильович___ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Денисюку Віталію Вікторовичу
(прізвище, ім’я, по батькові)

1. Тема роботи мобільний додаток системи управління дипломними проектами.

керівник роботи к.т.н., доц. Коваль Олександр Васильович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 2019р. № __

2. Строк подання студентом роботи 10 червня 2019 р.

3. Вихідні дані до роботи Мова програмування JavaScript, середовище розробки Visual Studio Code

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Провести аналіз предметної області, спроектувати архітектуру системи, розробити модуль “Викладач”, розробити модуль “Студент”, протестувати алгоритми роботи

5. Перелік ілюстративного матеріалу
Мета та завдання роботи, існуючі рішення, використані технології, архітектура системи, функції системи, архітектура проекту, алгоритм дій користувача, авторизація користувача, модуль “Викладач”, модуль “Студент”, експлуатаційні характеристики, висновки

6. Дата видачі завдання ”10”_жовтня_2018р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.18-15.01.19	
2	Розробка архітектури та загальної структури системи	15.01.19-04.02.19	
3.	Розробка структур окремих підсистем	05.02.19-11.02.19	
4.	Програмна реалізація системи	12.02.19-28.02.19	
5.	Оформлення пояснювальної записки	05.03.19-29.05.19	
6.	Захист програмного продукту	15.05.19	
7.	Передзахист	28.05.19	
8.	Захист		

Студент _____ _Денисюк В.В. _____
(підпис) (прізвище та ініціали,)

Керівник роботи _____ _Коваль О. В._____
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою роботи є створення мобільного додатку для системи управління дипломними проектами.

Мобільний додаток допомагає викладачам та студентам зручно слідкувати та керувати процесом написання дипломної роботи, створювати та обирати теми, надсилати та приймати заявки, ознайомлюватись з інформацією роботи кафедри.

У системі представлено дві ролі: “Викладач” та “Студент” для яких реалізовано різний функціонал. Розроблений продукт являє собою мобільний додаток для операційної системи Android.

Обсяг дипломної роботи складає 82 сторінки, 36 рисунки, 3 додатки.

ABSTRACT

The purpose of the work is to create a mobile application for a diploma project management system.

The mobile application helps teachers and students conveniently follow and manage the process of writing a thesis, create and select topics, send and receive applications, get acquainted with the information of the department's work.

The system presents two roles: "Teacher" and "Student" for which different functionalities are implemented. The product developed is a mobile application for the Android operating system.

The total amount of work is 82 pages, 36 drawings, 3 applications.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1. ЗАДАЧА РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ 10
2. АНАЛІЗ ПРОБЛЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ 12
2.1. Поняття дипломного проекту 12
2.2. Специфіка роботи над дипломним проектом 13
2.3. Зв’язок між студентом та дипломним керівником 14
2.4. Існуючі рішення 14
2.4.1. Atlassian JIRA 14
2.4.2. Trello 16
2.4.3. Порівняння існуючих рішень з запропонованим 16
3. ЗАСОБИ РОЗРОБКИ 18
3.1. Середовище розробки Visual Studio Code 18
3.2. Мова програмування JavaScript 19
3.3. Бібліотека React 20
3.4. Фреймворк React Native 23
3.5. Бібліотека для управління станом застосунку MobX та зв’язок з React 24
3.6. Емулятор мобільних додатків Genymotion Android Emulator 27
3.7. Операційна система Android 28
3.8. Онлайн-магазин мобільних додатків Google Play 30
4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 32
4.1. Архітектура програмної системи 32
4.2. Створення мобільного додатку за допомогою React Native 38
4.3. Архітектура проекту 39
5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ 43
5.1. Інсталяція та експлуатаційні характеристики 43

5.2. Робота користувача в системі 43
5.2.1. Екрани для ролі “Викладач” 46
5.2.2. Екрани для ролі “Студент” 50
ВИСНОВКИ 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 55
Додаток А 57
Додаток Б 59
Додаток В 74

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

СКБД — система керування базами даних;

API (англ. Application Programming Interface) — прикладний програмний інтерфейс;

БД — база даних;

Android — операційна система для мобільних пристроїв;

UX (англ. User Experience) — користувацький досвід;

DOM (англ. Document Object Model) — об’єктна модель документа;

ВСТУП

Онлайн-освіта — це одна з прогресивних форм навчання. Ця форма навчання має багато переваг, одна з яких полягає в економії часу на дорогу до навчального закладу. З’явилась можливість навчатися з дому або з будь-якого іншого місця.

Сьогодні існує проблема відсутності керування порядком роботи над дипломними проектами у вищому навчальному закладі (надалі ВНЗ) онлайн. Для прикладу студент повинен шукати кабінети, щоб дізнатися, як буде відбуватися проведення дипломного проекту, доступні теми, графік виконання, список дипломних керівників. Подібна ситуація існує у викладачів (дипломних керівників).

Внаслідок цього виникає безліч проблем та незрозумілих ситуацій, які не приємні як студентам, так і викладачам. Подібні проблеми відображаються на якості дипломного проекту, некомпетентності студента у важливих деталях, інформацією яких потрібно володіти на високому рівні.

Було поставлено завдання розробити систему управління дипломними проектами, а саме частину цієї системи – мобільний додаток на Android.

Розробленим додатком можуть користуватись як студент, так і викладач. Додаток розроблений для кращої та продуктивнішої роботи між студентами та викладачами над дипломним проектом.

Модуль студента надає функції пошуку тем для дипломної роботи, вибору та подачі заявки на кілька обраних тем. Також після того, як заявка оброблена викладачем, у профілі студента з’являється інформація про обрану тему дипломної роботи, дипломного керівника та графіку виконання дипломної роботи.

З іншого боку модуль викладача надає функції створення, редагування, видалення тем дипломних робіт, проектів, прийняття заявок від студентів, перегляд графіку дипломної роботи та можливість його редагування: виставлення статусів (виконано, на виконані, не виконано) та внесення вказівок до кожного етапу виконання дипломної роботи.

Також для обох користувачів додатку передбачена сторінка з інформацією про лабораторії кафедри, їхні напрями та викладачів, які входять до них. Кожен користувач має особисту сторінку з профілем, де знаходяться персональні дані: ПІБ, електрона пошта, телефон, фото тощо. Існує можливість редагування фото, електронної пошти, телефону.

Для розробки даного мобільного додатку було проаналізовано існуючі рішення, які призначені вирішувати проблеми управління проектами та запропоновано нове рішення, яке буде здатне вирішувати специфічні задачі управління дипломними проектами.

Записка містить 5 розділів.

У першому розділі описується задача вирішення проблеми управління дипломними проектами зі сторони студента та дипломного керівника.

У другому розділі описується аналіз проблеми зв’язку між студентом та дипломним керівником на період роботи над дипломним проектом.

У третьому розділі вказуються основні засоби розробки даної системи.

У четвертому розділі дано опис реалізованого програмного продукту та його архітектури.

У п’ятому розділі дано опис роботи користувача з програмною системою.

1. ЗАДАЧА РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ СИСТЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ

Метою роботи є створення мобільного додатку системи управління дипломними проектами для покращення розуміння етапів розробки дипломних проектів та зв’язку між студентами та викладачами.

Розроблена система повинна бути представлена як мобільний додаток операційної системи Android, яка є достатньо популярною серед користувачів.

Розроблена система повинна мати дві ролі – студент та викладач. Для кожної ролі повинні надаватись індивідуальні функції, які потрібні відповідним користувачам даної системи.

Користувач повинен мати можливість входу в систему під своїми обліковими даними (електронна пошта, пароль, роль). Також має бути можливість нагадування паролю, де користувач буде отримувати пароль на електронну пошту.

Модуль студента є важливою частиною мобільного додатку та має забезпечувати наступні можливості:

— пошук, перегляд та вибір теми дипломної роботи;
— подача заявки на кілька дипломних робіт;
— відміна поданих заявок;
— перегляд інформації по поточній дипломній роботі після підтвердження заявки викладачем;
— перегляд графіку виконання дипломної роботи;
— перегляд інформації по лабораторіям кафедри, їхнім напрямам та викладачам, які входять до неї;
— перегляд та можливість редагування профілю.

Модуль викладача є ключовою частиною даного додатку та має забезпечувати наступні можливості:

— перегляд, створення, редагування та видалення теми дипломної роботи;
— перегляд, створення, редагування та видалення проектів;

— прийняття заявок від студентів (бакалавр, магістр);
— зняття студента з теми дипломної роботи;
— перегляд графіку виконання дипломної роботи з можливістю редагування статусу кожного етапу та внесення певних вказівок, приміток;
— перегляд інформації по лабораторіям кафедри, їхнім напрямам та викладачам, які входять до неї;
— перегляд та можливість редагування профілю.

Мобільний додаток повинен взаємодіяти з наданим API, яке працює на базі Node.js. У випадку управління базою даних була обрана СКБД MySQL [9].

Додаток повинен підтримувати версії операційної системи Android починаючи з 6.0. Форми, кольори та інші елементи дизайну виконати на власний розсуд.

Додаток повинен дотримуватися усіх UX-законів та пройти тестування усіх можливих випадків – розгортання різних ситуацій.

Для написання додатку дозволено використовувати будь-які засоби розробки, які існують на сьогоднішній день.

2. АНАЛІЗ ПРОБЛЕМИ УПРАВЛІННЯ ДИПЛОМНИМИ ПРОЕКТАМИ

У даній роботі було проведено аналіз проблеми управління дипломними проектами та знайдено рішення для полегшення роботи над дипломним проектом як для студента так і для викладача.

2.1. Поняття дипломного проекту

Дипломний проект — кваліфікаційна робота, яка визначає рівень знань та умінь студента вирішувати нестандартні задачі та знаходити шляхи вирішення нових проблем.

Дипломні проекти (роботи) виконуються на завершальному етапі навчання студентів. Виконання дипломної роботи передбачає систематизацію, розширення теоретичних і практичних знань та застосування їх при вирішенні конкретних завдань, які можуть нести науковий, технічний та економічний характер.

Написання дипломної роботи є результатом вивчення студентами матеріалів, написання лабораторних робіт і здачі контрольних робіт з усіх дисциплін, які вивчались до цього моменту. Кожна дисципліна важлива та потребує уваги, адже кожна з них може дати інформацію, яка допоможе вирішити проблему специфічного характеру, якщо матеріал даної дисципліни був вивчений та закріплений на практиці.

Виконання дипломної роботи студентом показує його здатність працювати з літературою, фільтрувати інформацію та застосовувати її до проблем даної роботи.

Здатність висловити ідею продукту, який вирішує якесь завдання та запропонувати рішення проблеми є важливою ознакою підготовленості студента до праці у приватних компаніях та державних установах.

Завершальним етапом виступає захист дипломної роботи, де випускник має переконливо захистити свою роботу на відкритому засіданні державної екзаменаційної комісії.

2.2. Специфіка роботи над дипломним проектом

Студент, який випускається на останньому курсі свого навчання повинен продемонструвати свої вміння, знання та навички у своїй дипломній роботі, щоб отримати диплом бакалавра або магістра.

В першу чергу студент повинен обрати для себе дипломного керівника. У ролі дипломних керівників виступають викладачі. Після взаємного погодження студент разом з дипломним керівником висувають варіанти можливих тем, серед яких потрібно обрати одну. Завдання студента дослідити обрану тему, сформувати проблеми, які виникають у даній темі та запропонувати їх рішення.

Після обговорення проблематики даної теми з дипломним керівником, студент, у більшості випадках, відправляється на практику у фірму, яка займається подібними проблемами та проходить практику, де дізнається можливі рішення проблеми та працює над їх вирішенням.

Наступний етап представляє собою написання звіту по переддипломній практиці. В той же час, наприклад, для студентів спеціальностей, які пов’язані з комп’ютерними науками, визначається дата захисту програмного продукту, де студент повинен продемонструвати свої практичні навички, набуті на базі практики, представляючи рішення проблеми теми дипломної роботи у вигляді розробленого програмного продукту.

Записка є чи не найважливішою складовою дипломної роботи, адже там описується вся виконана робота над проблематикою теми. Тому цей елемент повинен бути зроблений до передзахисту та захисту дипломної роботи — наступних, заключних етапів роботи над дипломною роботою.

2.3. Зв’язок між студентом та дипломним керівником

Важливою складовою роботи над дипломним проектом є зв’язок між студентом та дипломним керівником, адже студент може дізнатися поради та настанови, які потрібно виконати для успішної здачі дипломного проекту.

Дипломний керівник виступає у ролі менеджера, який керує роботою студента протягом написання дипломної роботи, допомагає, нагадує та вказує на певні недоліки у роботі студенти. Недоліками можуть бути запізнення подачі відповідних документів, неправильність написання документів тощо.

Для вирішення проблеми управління дипломними проектами було запропоновано створити мобільний додаток, де як студент, так і дипломний керівник можуть слідкувати та керувати виконанням дипломної роботи.

Вся необхідна інформація по роботі над дипломним проектом подана у додатку. Тому можна з впевненістю сказати, що зв’язок між студентом та дипломним керівником стане кращим, оскільки вони зможуть вирішувати питання онлайн в рамках однієї платформи, де зібрано саме те, що необхідно для успішного виконання дипломної роботи.

2.4. Існуючі рішення

На даний момент універсальним програмним продуктом, який використовується в переважній більшості випадків для керування проектами є Atlassian JIRA. Також існує більш простіше рішення під назвою Trello.

2.4.1. Atlassian JIRA

Atlassian JIRA — система управління задачами у проектах [24], призначена для організації роботи над проектами та спілкування між співпрацівниками. Дана система розроблена компанією Atlassian в 2002 році. Зараз JIRA включає в себе три

проекти: JIRA Software (для розробників), JIRA Service Desk (підтримка проекту), JIRA Core (управління проектами), кожен з яких можна купити окремо. На даний момент (2019 рік) JIRA є однією з найпопулярніших систем управління задачами.

Система JIRA надає можливість створення проектів та можливість роботи над проектом за допомогою різних методів управління проектами. Серед відомих методів можна відзначити SCRUM.

Особливістю даного методу є процес, який дозволяє у фіксовані строки і невеликі по часу ітерації, які називаються спринтами, надати кінцевому користувачу працюючий продукт з новим функціоналом.

Головним елементом JIRA є дошки (англ. boards). Дошки включають в себе робочий процес, список задач, звітність. Також існує поняття задача (англ. issue), яка створюється адміністратором і додається у список задач (англ. backlog). Після цього формуються спринти та на дошці з’являються задачі, які потрібно виконати за поточний спринт.

Задачі переходять з одного етапу до іншого. Найпростішим набором етапів є такий: «заплановано» — «в роботі» — «зроблено». Задачі можуть нести різний характер, це може бути як просте завдання, так і завдання вирішення якогось багу у системі, що розробляється.

Також є індикатори важливості завдання, тобто якщо індикатором є червона стрілка, яка направлена вгору, то завдання вважається дуже важливим і його потрібно вирішити якомога швидше. І навпаки, синя стрілка, яка направлена вниз, означає, що задача на даний момент не є важливою.

Будь-які зміни в проблемі записуються в журнал активності. В JIRA представлено різноманітні діаграми, по яким можна визначити на якій стадії знаходиться проект, які проблеми наразі існують.

Усі ці елементи є важливими складовими управління проектами, запорукою яких є успішне виконання проекту.

2.4.2. Trello

Trello — безкоштовна багатоплатформна система управління проектами [25], розроблена Fog Creek Software.

Подібно до системи управління проектами та задачами JIRA, даний продукт надає можливість створення проекту та ведення його у стилі парадигми, відому як Kanban.

Проекти зображуються дошками зі списками. Кожен список містить картки-задачі, які можуть бути закріпленими за користувачем. Картки зображують задачі, вони містять назву задачі та короткий опис. Картки переходять з попереднього списку до наступного, таким чином, показуючи статус вирішення задачі. Картці може бути присвоєно відповідальних за неї користувачів. Користувачі та дошки можуть об'єднуватись в команди.

Станом на 2017 рік, підтримуються такі мобільні платформи, як iPhone та Android. Застосунок для iPad було випущено 12 березня 2013 року.

2.4.3. Порівняння існуючих рішень з запропонованим

Запропоноване рішення включає в себе рішення проблем, які пропонують існуючі рішення.

Для прикладу у розробленому додатку існує вкладка з графіком виконання робіт, пункти якого дипломний керівник може відмічати статусом(виконано, не виконано, на виконанні), до того ж залишати у формі певні вказівки, які допоможуть студенту зрозуміти суть проблеми, яку потрібно вирішити.

Серед переваг даного рішення проблеми теми дипломної роботи є поданий функціонал у додатку, який адаптований до специфіки роботи кафедри. Наприклад, є можливість ознайомлення з лабораторіями кафедри, робота з заявками як для студента, так і викладача.

Також є важливим відокремлення логіки роботи над дипломними проектами, введення графіку виконання дипломної роботи.

Існуючі рішення надають загальний функціонал керування проектами та частково вирішують проблему керування дипломними проектами, але керування

дипломними проектами має свою специфіку, вирішення проблеми якої може бути виконана в окремому програмному продукті.

Висновки до розділу

У даному розділі було проведено аналіз поняття дипломного проекту, специфіку роботи над проектами, зв’язок між студентами та викладачами на період виконання дипломного проекту. Також були розглянуті існуючі рішення та проведено порівняльну характеристику, де було визначено переваги даної системи.

3. ЗАСОБИ РОЗРОБКИ

При розробці програмного продукту важливим чинником є правильний вибір технологій та засобів розробки.

Інтерфейс користувача було реалізовано за допомогою фреймворка React Native на мові JavaScript в операційній системі macOS High Sierra та у середовищі розробки Microsoft Visual Studio Code. Додатковим засобом розробки виступив емулятор мобільних додатків Genymotion Android Emulator, який надає можливість тестування додатків на представленнях мобільних пристроїв операційної системи Android.

Також у розробці було використано знання бібліотеки React та було застосовано бібліотеку менеджменту стану застосунку MobX. Проведено аналіз операційної системи Android, визначено переваги та недоліки і зроблено остаточний висновок вибору розробки під дану операційну систему.

Джерелом дистрибуції даного додатку є магазин додатків Google Play.

3.1. Середовище розробки Microsoft Visual Studio Code

Microsoft Visual Studio Code — текстовий редактор для зручного написання коду програм [19]. Даний редактор розроблений компанією Microsoft для операційних систем Windows, Linux та macOS. Вважається “легковаговиком” серед потужних середовищ розробки програмного забезпечення таких як IntelliJ IDEA та WebStorm.

Перевагою Visual Studio Code над подібними рішеннями є швидкість роботи, простий та лаконічний інтерфейс.

Редактор містить вбудований відладник, інструменти для роботи з системою контролю версій Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки.

Також редактор має магазин розширень, їх можна встановлювати безкоштовно. Розширення допомагають покращити роботу з кодом. Доступні розширення для багатьох мов програмування.

Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, PHP, Python, HTML, CSS та інші.

3.2. Мова програмування JavaScript

Мова програмування JavaScript (JS) — прототипна, скрипкова (підмножина об'єктно-орієнтованої) мова програмування з динамічною типізацією [1]. Реалізація стандарту ECMAScript [10]. Вона була винайдена Бренданом Ейхом, співзасновником проекту Mozilla, the Mozilla Foundation, та Mozilla Corporation.

Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

До того ж мова JavaScript використовується для:

— створення односторінкових веб-застосунків (React, AngularJS, Vue.js);

— програмування на стороні сервера (Node.js);

— мобільних застосунків (React Native [11], Cordova);

Розпочати писати код на JavaScript дуже легко, просто створивши файл з розширенням .js, щоб запустити програму, потрібно мати встановлений Node.js, який запускає код [6].

Мова надає багато можливостей для розробки різноманітних елементів веб-сторінок: “слайдери”, створення анімацій, випливаючі меню та інші динамічні елементи сторінки.

Крім розробки функціоналу для динаміки на веб-сторінці, мова JavaScript дозволяє писати різноманітні API. Для цього використовується Node.js та фреймворки Express, Koa, NestJS тощо.

Мова JavaScript [16] має C-подібний синтаксис, але в порівнянні з мовою Сі

має деякі корінні відмінності, з яких:

— об'єкти, з можливістю інтроспекції і динамічної зміни типу через механізм прототипів функції як об'єкти першого класу обробка винятків

— автоматичне приведення типів

— автоматичне прибирання сміття

— анонімні функції

Деякі вважають, що JavaScript — це Java, але це помилка. Ці мови є об’єктно-орієнтованими і використовуються в клієнтських веб-додатках. Серед відмінностей можна відмітити:

— Java має статичну типізацію, натомість JavaScript має динамічну типізацію [18];

— Java реалізує ООП підхід, який заснований на класах, JavaScript — на прототипах;

— Java завантажується з скомпільованого байт-коду, коли JavaScript інтерпретується прямо з файлу.

Сьогодні мова JavaScript [17] підтримується у всіх сучасних браузерах. Але у деяких браузерах є моменти, коли функції ES6 не працюють, для прикладу функція includes у браузері Internet Explorer 11. Щоб ця функція працювала, необхідно використовувати поліфіли, які перетворюють код з нових ES6/7/8 [14] на стандартний ES5, усі функції якого підтримуються даним браузером.

Для того, щоб використовувати функції нових специфікацій ECMAScript потрібно використовувати Babel — транспілятор для JavaScript, який перетворює “новий” код з ES6 на ES5 [15].

3.3. Бібліотека React

Бібліотека React [13] дозволяє створювати як малі так і великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. Мета бібліотеки полягає в тому, щоб бути швидкою, простою та масштабованою. Бібліотека React обробляє тільки інтерфейс користувача у веб-застосунках. Це відповідає вигляду у шаблоні модель-вид-контролер (MVC).

Особливістю бібліотеки є віртуальний DOM. Це дозволяє бібліотеці визначити, які частини реального DOM змінилися, порівняно (diff) зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера (рисунок 3.1).

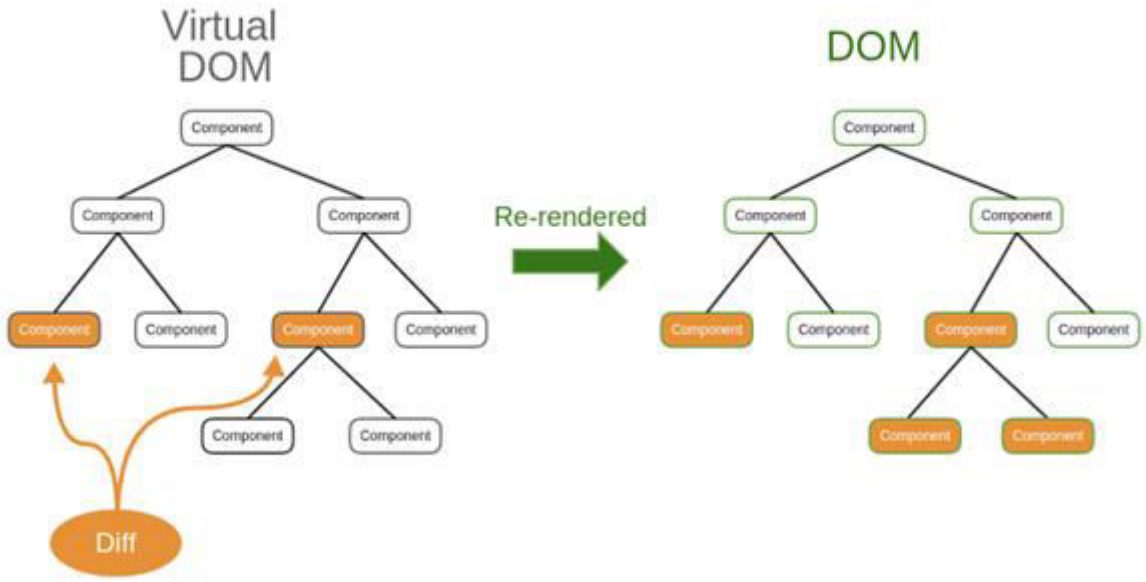

Рисунок 3.1 — Схема роботи Virtual DOM з справжнім DOM

Тому здається, що при змінах розташування елементів, оновлюється вся сторінка, але це не так, саме за допомогою Virtual DOM оновлення відбувається частково для тих елементів, які змінили свій стан.

Компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React [5]. JSX надає ряд атрибутів елементів, призначених для відображення тих, що надаються у форматі HTML.

Користувацькі атрибути можуть бути передані компоненту. Всі атрибути будуть отримані компонентом як реквізит. Це можуть бути структури даних, функції та інші компоненти.

Компонент бібліотеки React [20] має свій життєвий цикл та методи, які відносяться до нього. В процесі роботи компонент проходить через ряд етапів життєвого циклу.

На кожному з етапів викликається певна функція, в якій ми можемо визначити будь-які дії. Детальний шлях життєвого циклу, усіх методів життєвого циклу можна навести у невеликій схемі на рисунку 3.2.

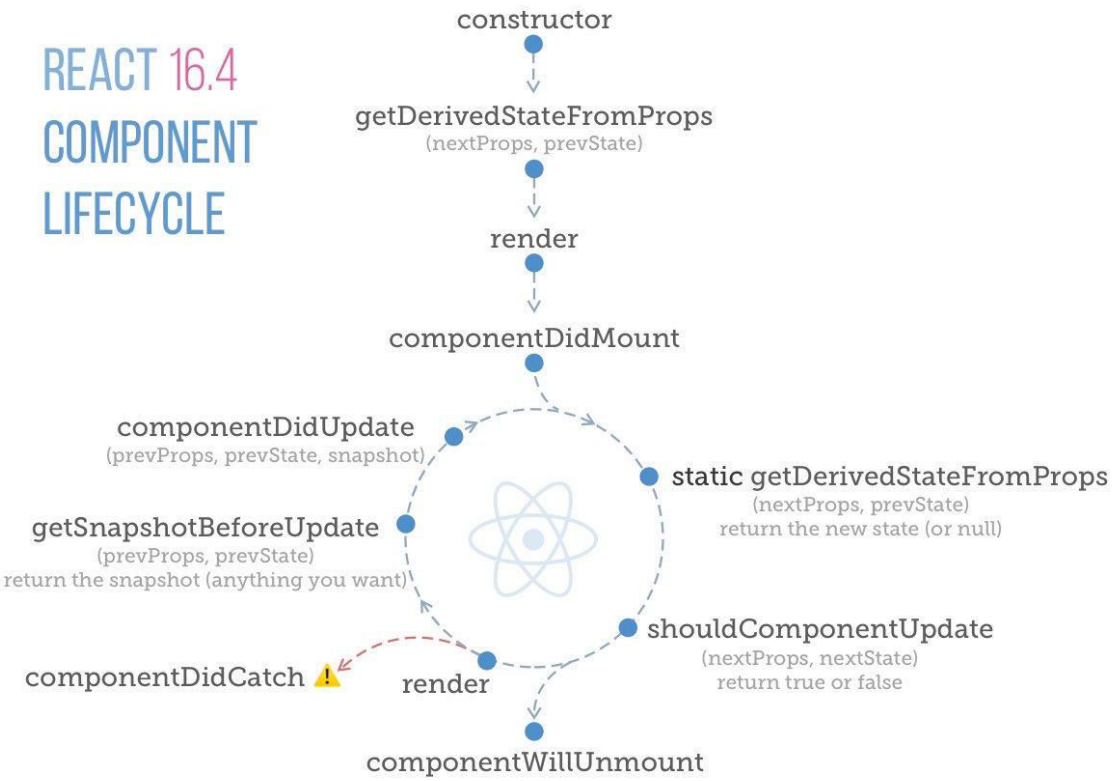

Рисунок 3.2 — Життєвий цикл React-компонента

На сьогоднішній день бібліотека React широко використовується у розробці веб-інтерфейсів, вона є популярною. Завдяки великій спільноті однодумців та користувачів даної бібліотеки, вона стає кращою з кожним днем.

Проаналізувавши подібні бібліотеки для розробки веб-інтерфейсів користувача та зважаючи на переваги бібліотеки React було прийнято рішення використати даний інструмент для написання програмного продукту, адже дана бібліотека надає можливості швидко розробити інтерфейс та надалі легко масштабувати застосунок.

3.4. Фреймворк React Native

React Native — це фреймворк реалізації інтерфейсу мобільних додатків [8] з відкритим вихідним кодом, створений Facebook. Він використовується для розробки додатків для Android, iOS і UWP, тобто при написанні одного додатку, він буде працювати на усіх платформах. Фреймворк [12] дозволяє розробникам використовувати бібліотеку React разом з власними можливостями платформи [3].

Проведемо порівняння між нативною, гібридною розробкою мобільних додатків та розробку з фреймоврком React Native (рисунок 3.3).

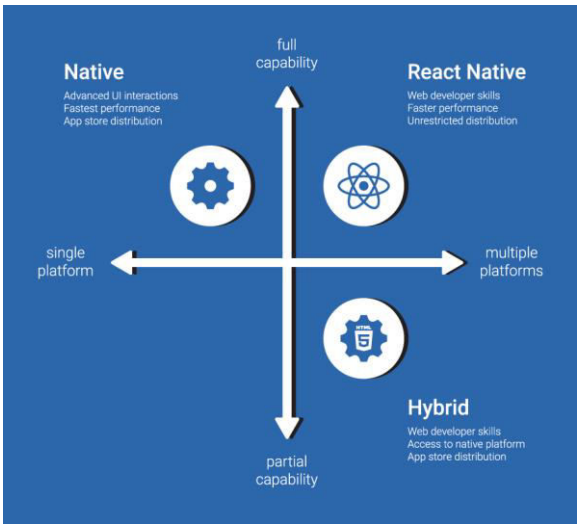

Рисунок 3.3 — Порівняння варіантів розробки мобільних додатків

Фреймворк React Native [21] використовує подібні принципи бібліотеки React [7], за винятком того, що React Native не маніпулює DOM через Virtual DOM. Вона виконується у фоновому процесі безпосередньо на кінцевому пристрої і зв'язується з власною платформою через серіалізацію, асинхронний і пакетний Bridge.

React Native не використовує HTML або JSX. Замість цього повідомлення з потоку JavaScript використовуються для маніпулювання нативними переглядами.

3.5. Бібліотека для управління станом застосунку MobX та зв’язок з React

Ключовими атрибутами React-застосунку є його компоненти та дані, які зберігаються у них у об’єкті state. Компоненти бувають батьківськими та дочірніми.

Зв’язати дані між цими компонентами дозволяє об’єкт props, куди можуть передаватись дані, функції та інші компоненти.

Потік даних є однонаправленим — від батька до сина (рисунок 3.4). Це допомагає компонентам бути простими та передбачуваними.

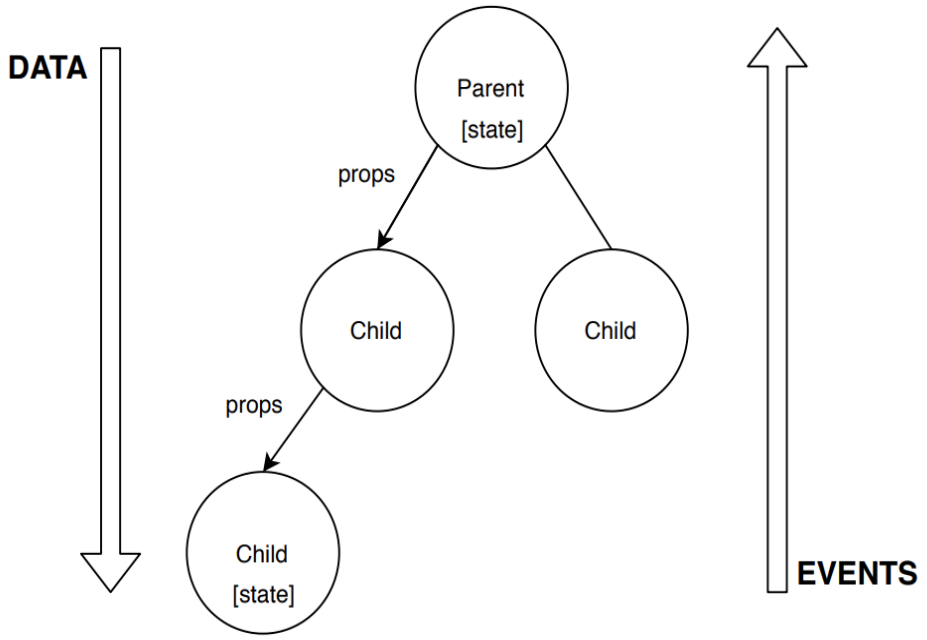

Рисунок 3.4 — Представлення потоку даних та подій у React-застосунку

Також існують події, які можуть викликати зміни у батьківських компонентах, що також передаються через props.

У великих застосунках існує проблема відправлення та отримання даних з одного компоненту в інший. Передача даних через props є допустимою, але вона не завжди є зручною, оскільки компоненти бувають вкладені один в одний і тоді доводиться передавати дані в кожному компоненті — зверху вниз.

Підтримувати надалі такий флоу буде складно. Тому було винайдено бібліотеки менеджменту стану застосунку. Прикладами таких бібліотек є Redux та MobX [22], оскільки я використовував MobX, то далі мова піде про неї.

Бібліотека MobX керує станом застосунку та оброблює його як електронну таблицю.

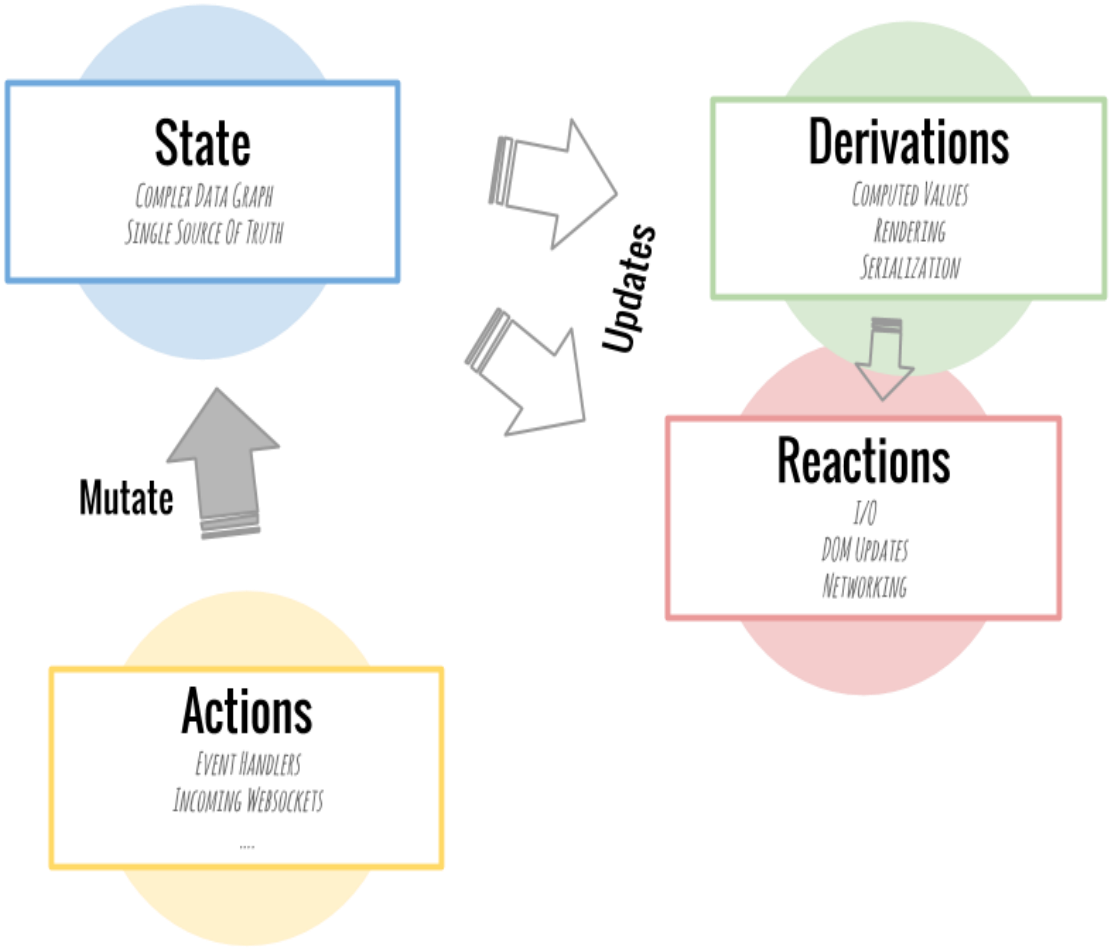

Рисунок 3.5 — Елементи бібліотеки MobX та їхня взаємодія

Бібліотека представляє схему в яку входять такі елементи (рисунок 3.5) як:

— стан застосунку (англ. State). Графи об’єктів, масивів, посилань, які формують модель застосунку.

— похідні (англ. Derivations). Це значення, яке може бути обчислено автоматично з даних стану застосунку.

— реакції (англ. Reactions). Вони перевіряють чи оновився DOM або мережеві запити виконались успішно.

— дії (англ. Actions). Дії змінюють стан застосунку. Бібліотека прослідкує, щоб всі зміни в стані застосунку, які були викликані діями, автоматично обробились усіма похідними та реакціями синхронно.

Бібліотека MobX використовує шаблон проектування “Наглядач” (Observer), схема якого зображена на рисунку 3.6.

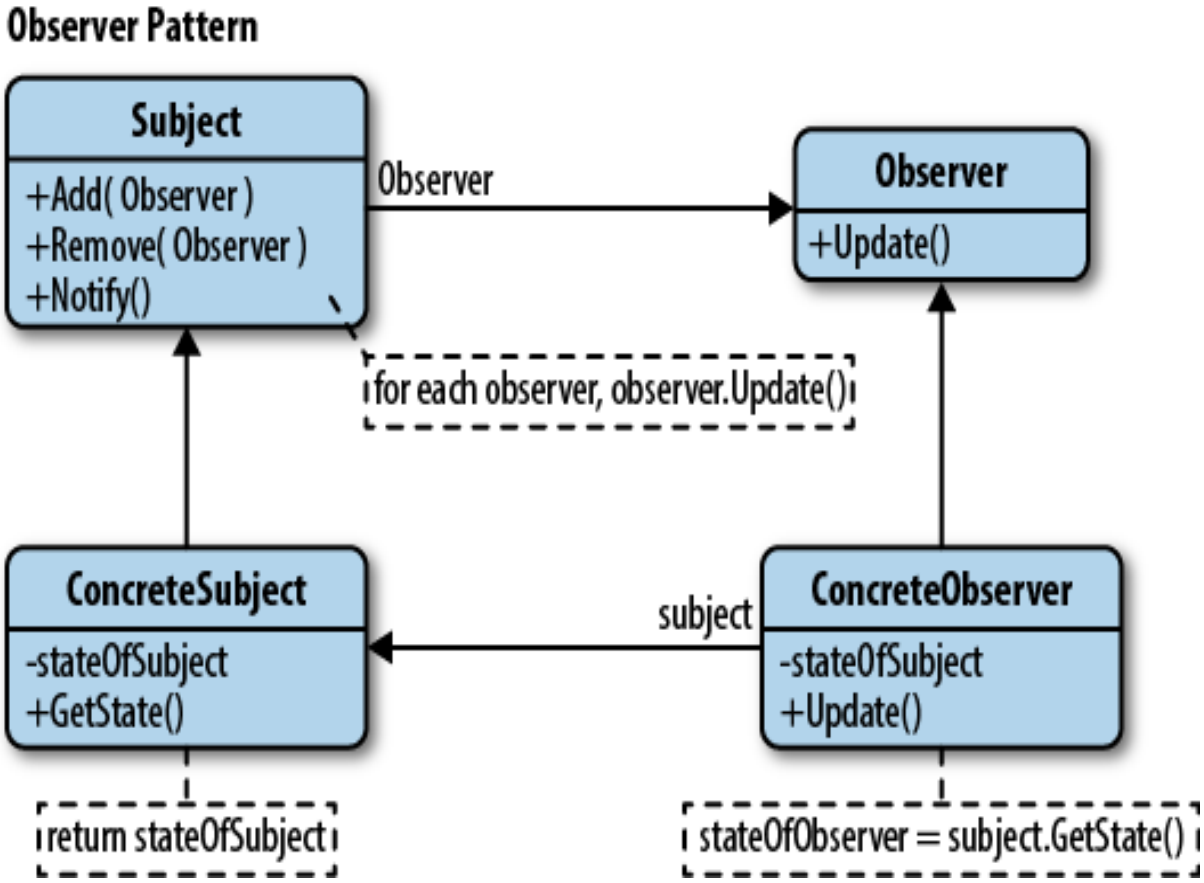

Рисунок 3.6 — Схема роботи паттерну Observer

Саме перевагам цього шаблону робота з сховищами даних є простою та зручною. Наглядачами виступають компоненти, а даними є наглядуваними (observable). При виконанні дії, об’єкт з даними, що є observable може змінюватись і ця зміна відобразиться у компоненті.

Історія MobX обертається навколо observables. Дії мутують ці observables. Похідні та реакції спостерігають і реагують на зміни цих observables. Наглядуючі, дії і реакції утворюють основну тріаду.

Цикл подій у MobX можна зобразити наступним чином (рисунок 3.7):

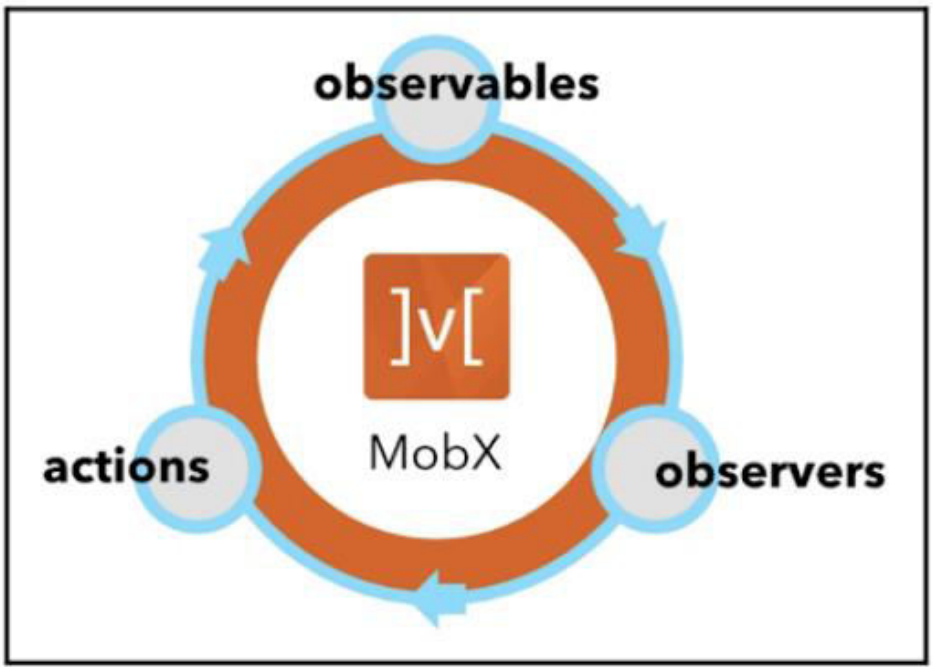

Рисунок 3.7 — Цикл подій у бібліотеці MobX

Враховуючи усі переваги даної бібліотеки, можна сказати, що вибір був правильним, адже використавши MobX на практиці, застосунок отримав велику продуктивність. З цим підходом можна легко і швидко маніпулювати даними та отримувати їх з потрібних частин застосунку.

3.6. Емулятор мобільних додатків Genymotion Android Emulator

Емулятор Genymotion Android Emulator — це емулятор операційної системи Android [9], який містить повний набір датчиків і функцій для взаємодії з віртуальним середовищем Android. Genymotion [23] надає можливість перевірки програми для Android на широкому діапазоні віртуальних пристроїв для розробки, тестування і

демонстраційних цілей. Інтерфейс програми є простим та зрозумілим (рисунок 3.8).

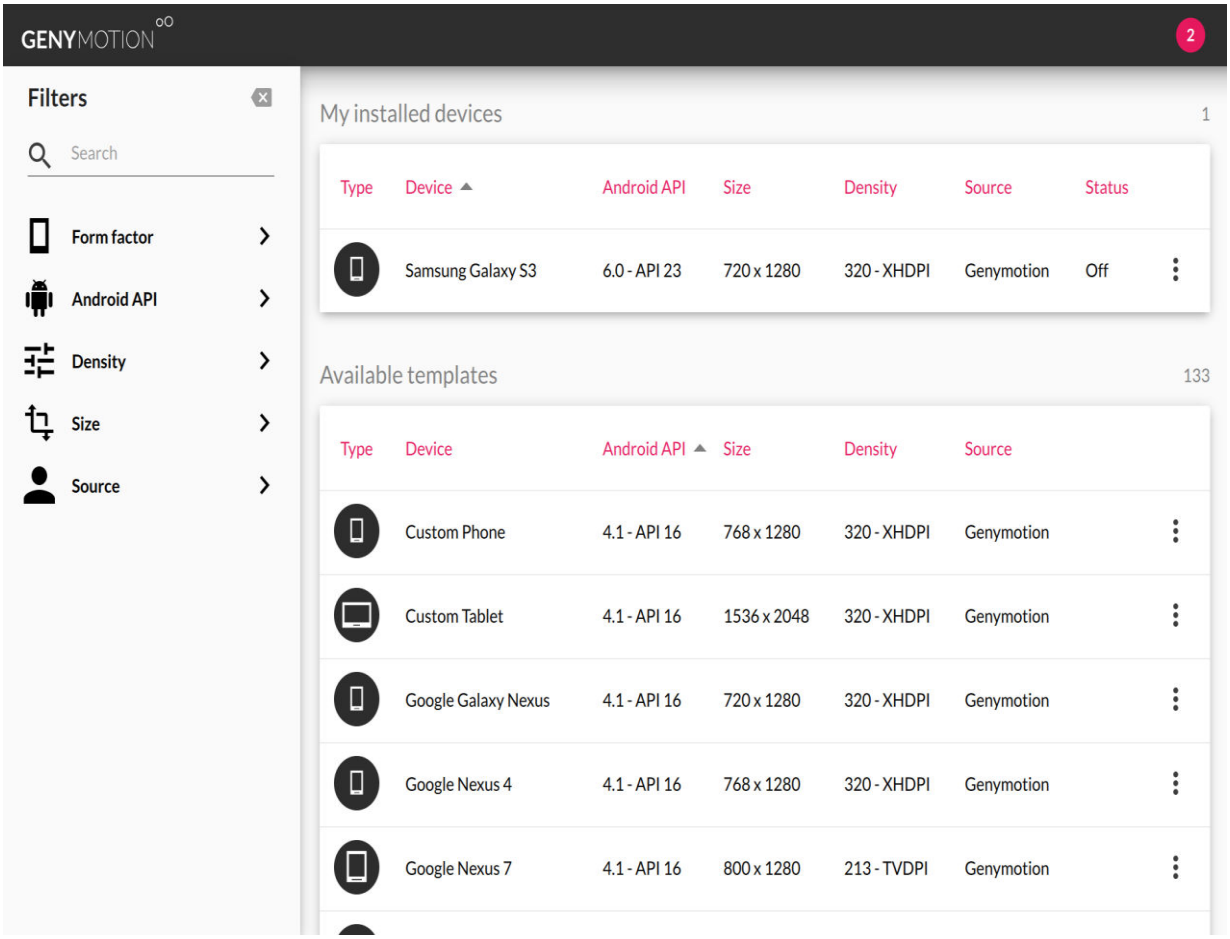

Рисунок 3.8 — Інтерфейс користувача Genymotion Android Emulator

Можемо завантажити та запустити будь-який пристрій з представлених. Genymotion є швидким, простим у встановленні і потужним завдяки зручним для користувача сенсорам віджетів і особливості взаємодії. Він доступний для операційних систем Windows, MacOS і Linux.

3.7. Операційна система Android

Операційна система використовується у мобільних пристроях, планшетах, електронних книгах, наручних годинниках тощо. Розробником даної операційної системи є Google. Операційна система побудована на ядрі Linux і власній віртуальній машині Java від розробників Google.

Цікавим фактом є той, що у 86 відсотків смартфонів, які продали у другому кварталі 2014 року, була встановлена операційна система Android.

Перевагою Android над iOS є те, що вона є відкритою платформою, що надає необмежені можливості розробникам.

Операційна система дає можливість встановлення застосунків з офіційного репозиторія Google, який надає найбільшу в світі базу програм. Така особливість існує тому, що кожний розробник може самостійно написати програму для мобільного пристрою і розмістити її в магазині Google Play.

Варто відмітити, що додатки на пристрої, у яких операційна система Android, можуть бути встановлені і через комп'ютер шляхом завантаження файла .apk і його установки на мобільному пристрої.

Життєвий цикл застосунку на Android наведено нижче на рисунку 3.9.

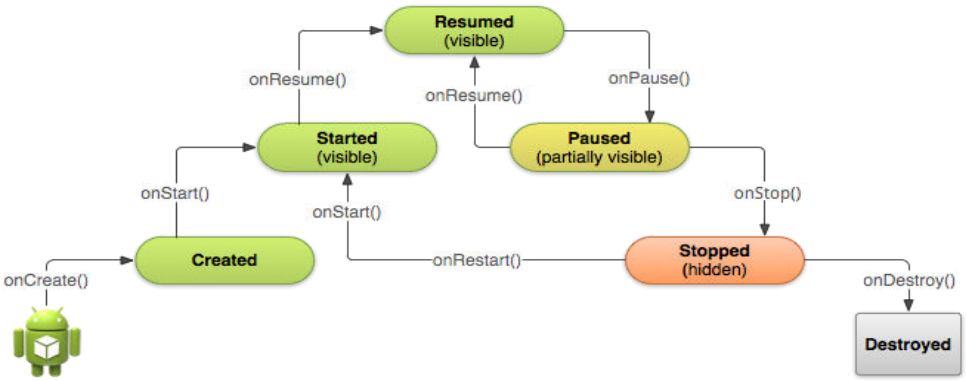

Рисунок 3.9 — Життєвий цикл застосунку на Android

Однією з особливостей системи Android є та, що застосунок може запустити компонент іншої програми. Наприклад, якщо ви хочете надати користувачеві можливість фотографувати, використовуючи камеру пристрою, то, оскільки напевно є інший додаток, яке може виконати цю дію, замість того щоб розробити операцію фотографування в своєму додатку, ви можете викликати такий додаток. Вам не потрібно впроваджувати код з програми для камери або навіть встановлювати на нього посилання.

Замість цього ви можете просто запустити операцію фотографування з програми для камери. По завершенні цієї операції фотографія буде повернута в ваш

додаток, і її можна буде використовувати. Для користувача це буде виглядати як один додаток.

Одним з недоліків цієї операційної системи є проблема сумісності. Нові версії системи конфліктують зі старими пристроями, це призводить до погіршення стану батареї, усілякі зависання, перезавантаження. В цьому плані система iOS показує себе краще.

Але не зважаючи на усі переваги та недоліки даної операційної системи, можна з упевненістю сказати, що дана система буде популярною і надалі, адже вона є лояльною та більш доступною для користувачів. Тому було зроблено рішення зосередитись на розробці саме під цією операційною системою.

3.8. Онлайн-магазин мобільних додатків Google Play

Даний продукт від компанії Google надає можливість скачування та встановлення мобільних додатків, ігор, офісних програм тощо. Більшість додатків є безкоштовними для скачування.

Сервіс надає можливості пошуку по категоріям, популярності та кількості скачувань. Розробники понад 150 країн можуть завантажувати свої програми на Google Play, але не кожна з них підтримує реєстрацію продавця. Кількість скачувань додатків досягло 82 мільярдів.

У магазині зустрічаються як корисні програми так і не потрібні, які можуть містити в собі віруси, після чого пристрій починає погано працювати. Також є певний відсоток неякісних додатків, які мають негативні відгуки та коментарі від користувачів платформи.

Дана платформа є основним інструментом дистрибуції програмних продуктів, а саме мобільних додатків.

Висновки до розділу

У даному розділі було представлено засоби розробки. Обраним текстовим редактором став Microsoft Visual Studio Code, який є простим та швидким у використанні і надає багато можливостей для розробки програмного продукту.

Мовою програмування було обрано JavaScript, оскільки дана мова дозволяє гнучко описувати виконання дій користувача, алгоритмів, має Сі-подібний синтаксис та є потужною мовою у розробці користувацьких інтерфейсів.

Було обрано бібліотеку для розробки React, у результаті чого було обрано фреймворк для розробки інтерфейсів мобільних додатків React Native, оскільки принципи роботи React схожі з React Native. Перевагами даного фреймворку є розробка мультиплатформених додатків, швидке виконання та опціональні знання нативних мов для розробки мобільних додатків Java для Android та Swift для iOS.

Для керування стану програми було обрано бібліотеку MobX. Дана бібліотека використовує патерн проектування “Наглядач” (Observer) та надає багато можливостей для зручного та швидкого керування станом застосунку, що робить застосунок спроможним до масштабованості.

Додатковим інструментом розробки став емулятор мобільних пристроїв Genymotion Android Emulator, який надає змогу тестування розроблювальних мобільних додатків на різних пристроях та версіях Android.

Було розглянуто можливості операційної системи Android, її особливості, переваги та недоліки. Дана система є популярною серед користувачів.

Похідним пунктом став огляд магазину застосунків для Android від компанії Google — Google Play.

Було вивчено особливості завантаження і розгортання додатку на даній платформі для дистрибуції програмного продукту.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Мобільний додаток управління дипломними проектами повинен мати функціонал авторизації користувача, функцію нагадування паролю, система має містити дві ролі — студент та викладач.

Модуль викладача повинен виконувати функції створення, редагування, видалення теми дипломної роботи, прийняття заявок від студентів, введення графіку виконання дипломної роботи, перегляд інформації по лабораторіям, перегляд та редагування профілю.

Модуль студента, крім останніх двох названих функцій вище, повинна мати функції перегляду доступних тем для дипломних робіт, подачі заявок на обрані теми, перегляд графіку виконання дипломної роботи.

Для реалізації всіх поставлених задач була обрана компонентна архітектура [4], яка є основним паттерном розробки додатків на React та React Native [3].

4.1. Архітектура програмної системи

Правильно закладена архітектура програмної системи є результатом успішно розробленого продукту. Оскільки мобільний додаток є лише частиною всієї системи, до якої входять API, яке надає методи створення, редагування та видалення об’єктів даних. API написано на платформі Node.js, для кращої роботи з кодом було використано веб-фреймворк Express.js.

Веб-фреймворк Express.js представляє широкий набір функцій для мобільних та веб-додатків. Також фреймворк дозволяє використовувати нативні функції Node.js, не конфліктуючи з тими, що надає фреймворк.

Система базуватиметься на RESTful архітектурі, що дозволить легко відокремити клієнтський додаток від програмних інтерфейсів для роботи з базою даних.

Серед переваг даної архітектури є простота уніфікованого інтерфейсу, прозорість зв'язків між компонентами системи для сервісних служб, переносимість компонентів системи шляхом переміщення програмного коду разом з даними. Побудована модель клієнт-сервер, тому частини системи працюють самостійно.

У ролі системи керування базою даних було обрано MySQL. Це реляційна СКБД, що підтримує структуровану мову запитів SQL і може застосовуватися у ролі SQL-сервера. Серед переваг можна відзначити безпечність, швидкодію, надійність та переносимість.

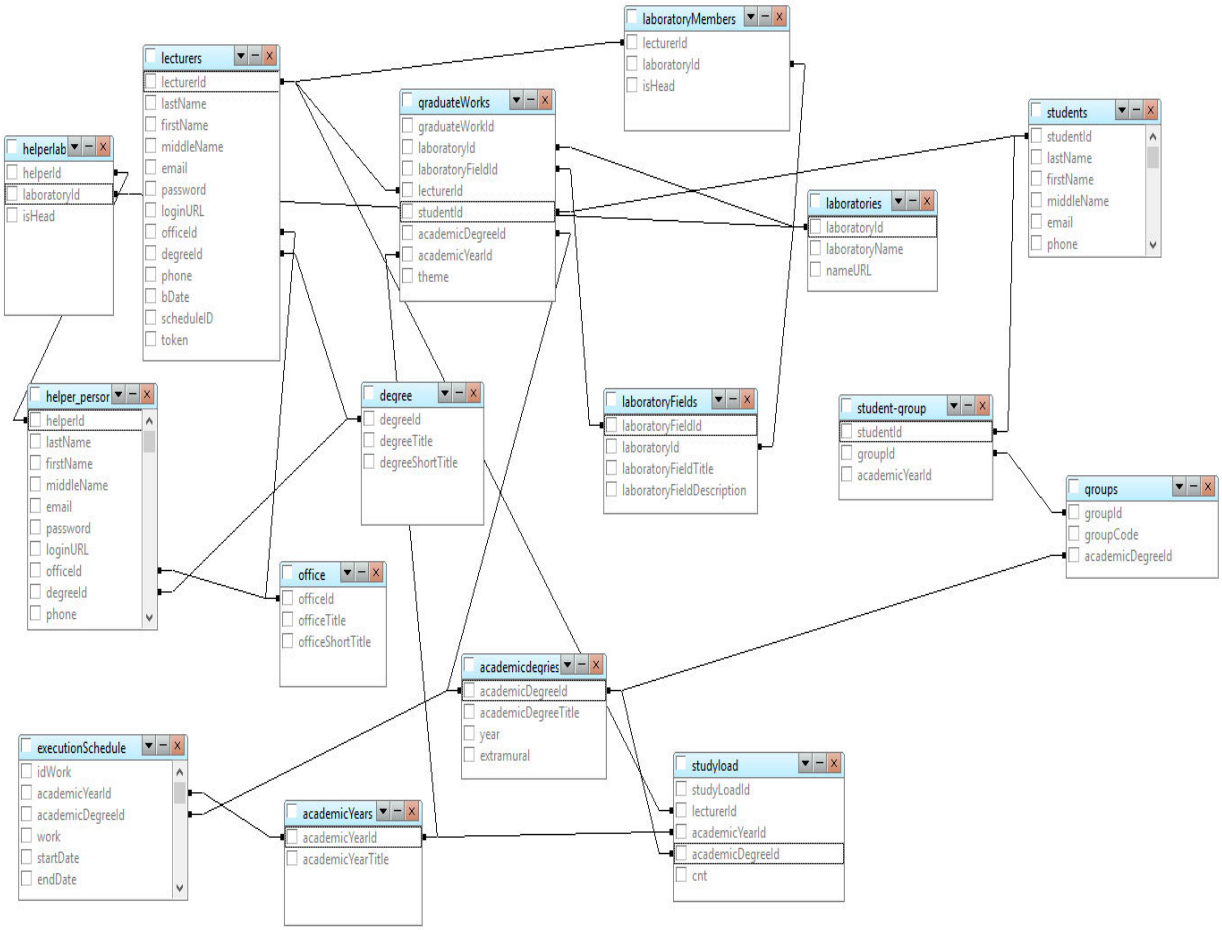

Рисунок 4.1 — Структура бази даних

Дана система має доволі складну структуру бази даних, таблиці якої зображено на рисунку 4.1.

Також іншою частиною системи є веб-додаток, який надає подібний функціонал, але у виконані веб-сайту. Веб-додаток написаний за допомогою бібліотеки React на мові програмування JavaScript.

Він також містить роль студента та викладача, до того ж, існує що додаткова роль, яка називається “Доп. персонал”. Цій ролі надано більше можливостей, ніж викладачу.

Це користувачі, які слідкують за роботою систему та кафедри, створюють та редагують графіки виконання дипломних робіт, додають та видаляють викладачів. Ця роль несе собою функції супер-адміна, який повністю володіє системою та може виконувати функції, які заборонені для виконання іншим ролям системи.

Зв'язок між усіма вузлами та компонентами проектованої системи відображено на рисунку 4.2.

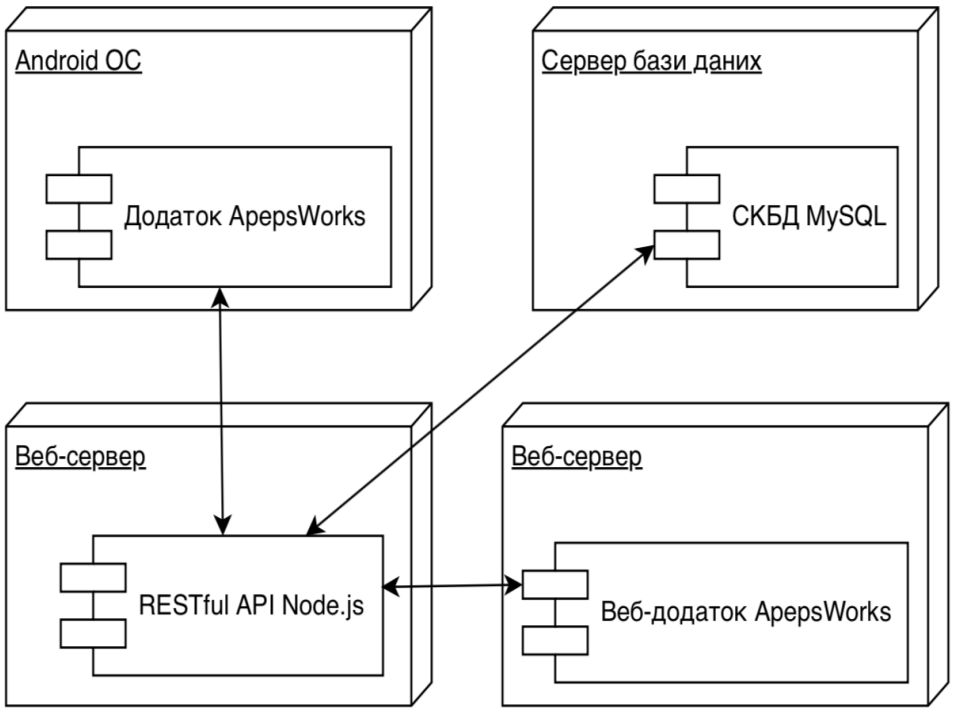

Рисунок 4.2 — Зв’язок між вузлами та компонентами проектованої системи

Розглянемо детальніше архітектуру самого мобільного додатку, моделі та функціонал, який реалізовано для двох ролей користувачів.

Спочатку розглянемо діаграму прецедентів модуля “Викладач”, оскільки даний модуль має елементи функціоналу адміністратора, зо зображено на рисунку 4.3.

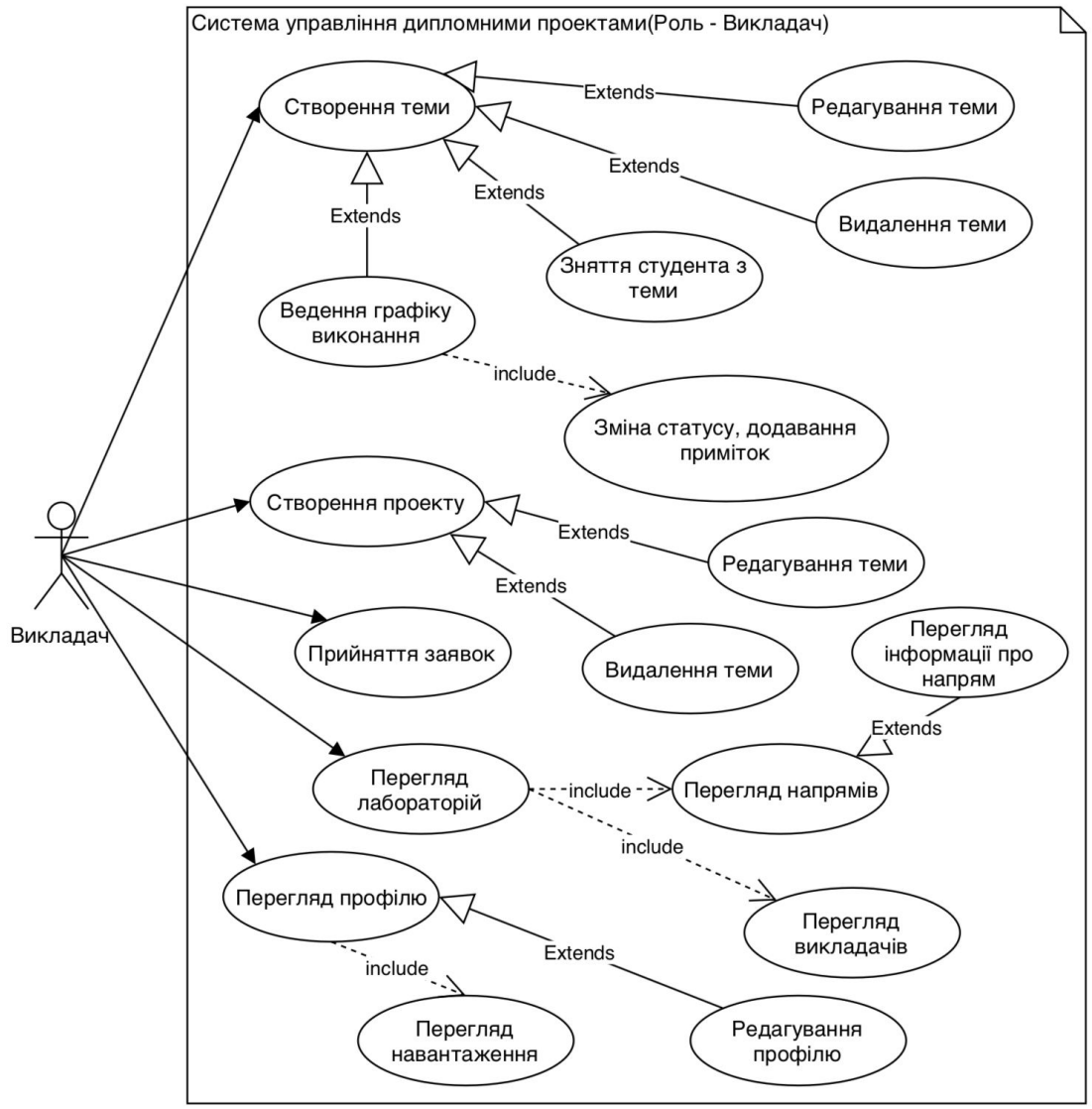

Рисунок 4.3 — Діаграма прецедентів модуля “Викладач”

Як вже було сказано, викладач має розширені можливості, яка надає система. Усі прецеденти вже були описані раніше і їхню роботу можна буде побачити у наступному розділі.

Також користувачем системи є студент. Студент має обмежені можливості, які надані системою, але не зважаючи на це, роль студента також є важливою, адже

дипломна робота пишеться саме студентом. Тому його функціоналу було приділено не менше уваги, ніж функціоналу для викладача.

Обидві ролі мають спільні прецеденти, наприклад, перегляд та редагування профілю. Детальніше прецеденти студента можна побачити на рисунку 4.4.

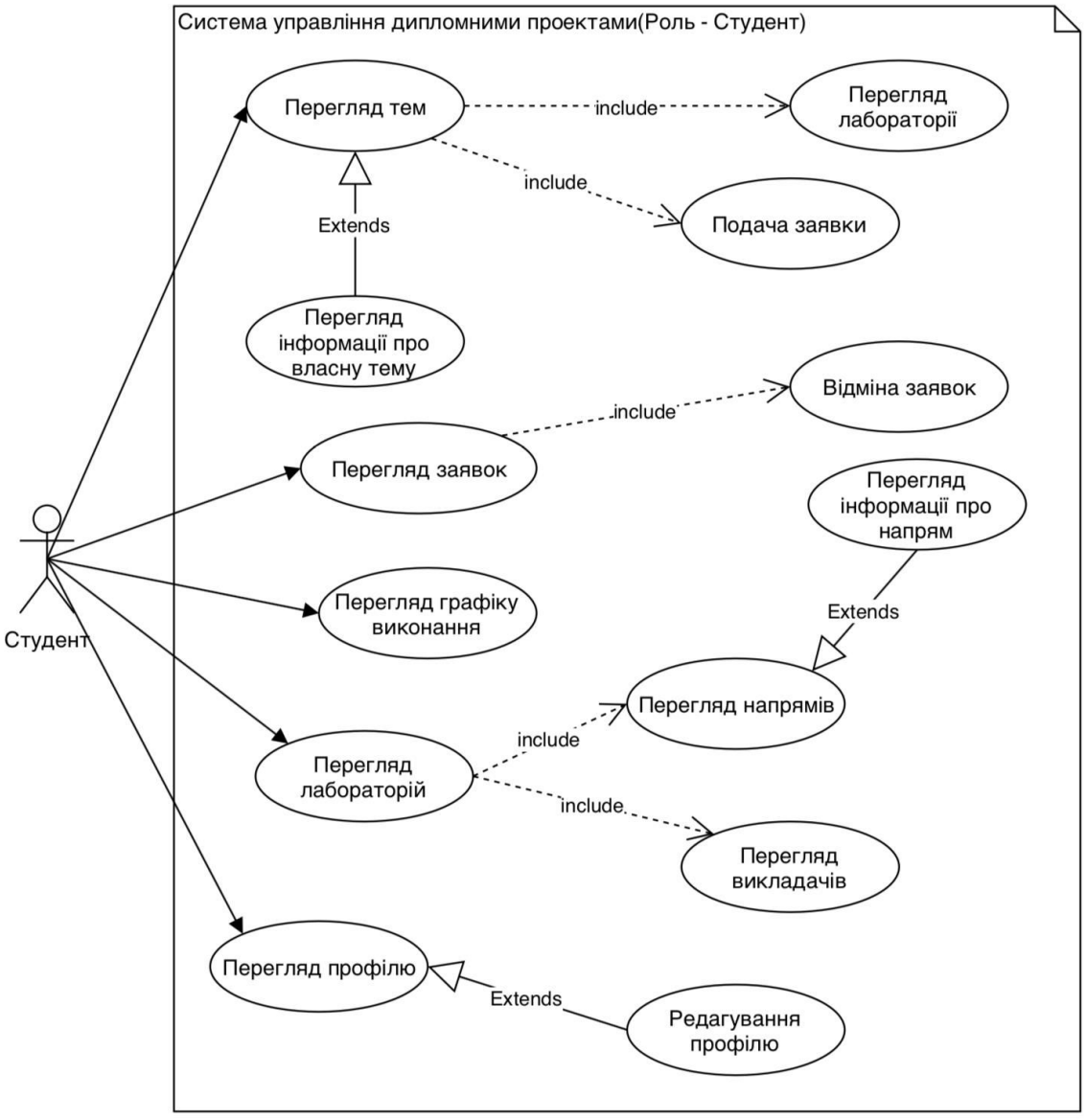

Рисунок 4.4 — Діаграма прецедентів модуля “Студент”

Алгоритми створення, редагування, видалення тем, заявок та інших об’єктів системи мають спільні методи та дії. Наприклад, для того, щоб виконати авторизацію користувача у систему необхідно зібрати дані з форми, у яку

користувач ввів свої дані, в один об’єкт. Після цього викликати метод, який робить запит на відповідну адресу та пересилає дані, які ми зібрали.

Коли сервер отримує дані, він виконує роботу авторизації, в результаті чого повертає на клієнтську частину токен з об’єктом користувача, де знаходиться його інформація. Цей алгоритм можна зобразити на діаграмі послідовностей (рисунок 4.5), що влучно підходить для опису подій.

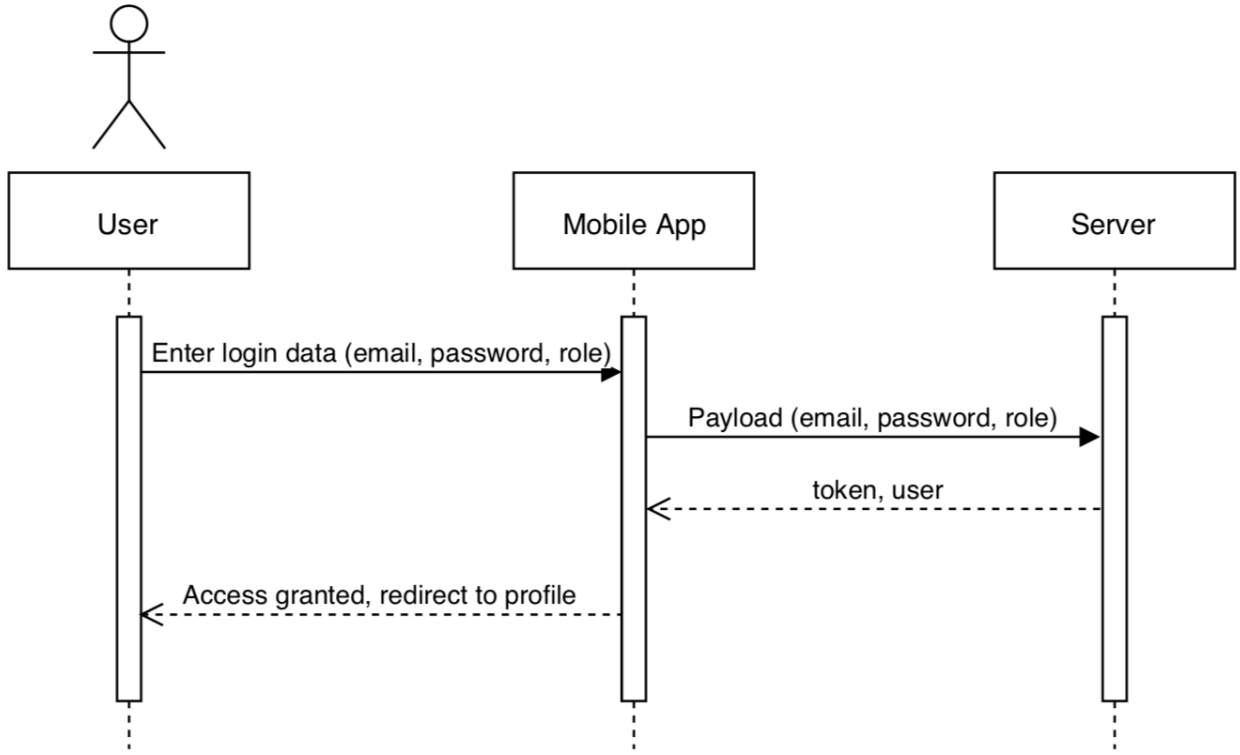

Рисунок 4.5 — Діаграма послідовності для функції “Авторизація”

Подібний алгоритм застосований до усіх функції, які маніпулюють даними, як зі сторони викладача, так і зі сторони студента, за винятком того, що при наступних запитах обов’язково потрібно слати токен та роль користувача, що повернувся нам від сервера при авторизації.

4.2. Створення мобільного додатку за допомогою React Native

Для запуску проекту необхідно скористатися менеджером пакетів (npm), які входять в склад Node.js [2]. Перед цим необхідно мати встановленим Node.js.

Після цього, переконавшись, що node.js та npm встановлено (це можна перевірити ввівши команду “node -v && npm -v” у терміналі) можемо встановити React Native CLI (інтерфейс командного рядка) завдяки якому надалі буде розгорнуто проект, який готовий до роботи. Для цього виконуємо команди “npm install -g react-native-cli”.

Також перед безпосередньою роботою над створенням проектом потрібно мати встановлений Java Development Kit 8.

Результатом команди “react-native init ProjectName” буде створений проект у директорії, з якої виконували відповідні команди.

Перед запуском проекту потрібно відкрити емулятор, у якому буде запущена програма, для цього користуємось Genymotion Android Emulator (рисунок 4.6).

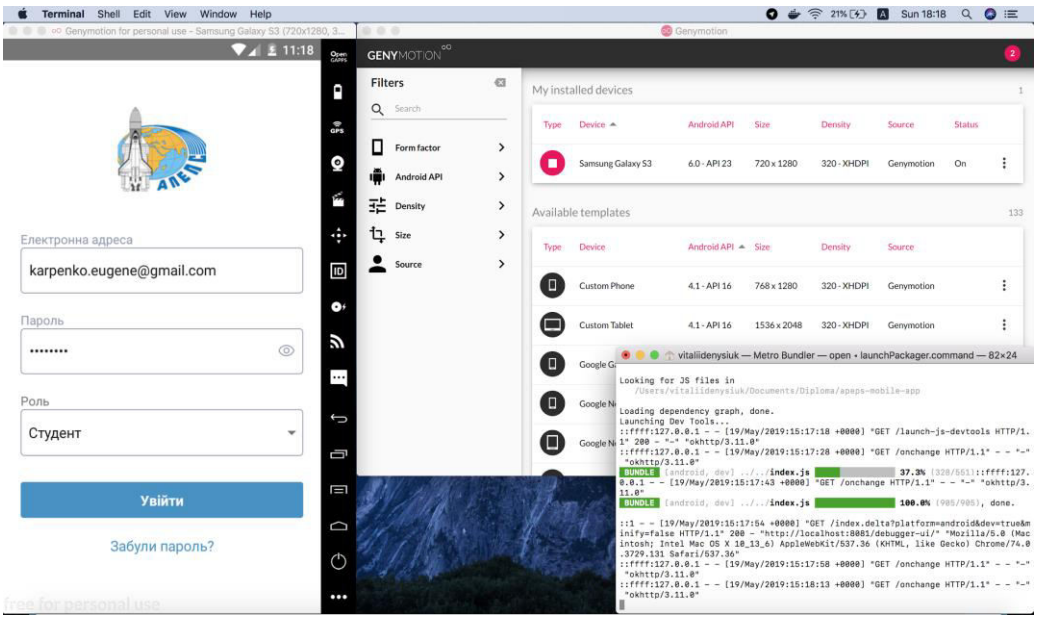

Рисунок 4.6 — Зразок запущеного проекту на React Native

Для того, щоб відкрити та запустити проект потрібно виконати наступні команди — “cd ProjectName”, “react-native run-android”.

В результаті виконання однієї команди був запущений сервер, відбулось збирання файлів та сформувався граф залежностей.

4.3. Архітектура проекту

При розробці проекту важливо підтримувати чітку та прозору архітектуру, структуру директорій та файлів. Нижче наведено список директорій, які входять до даного проекту (рисунок 4.7).

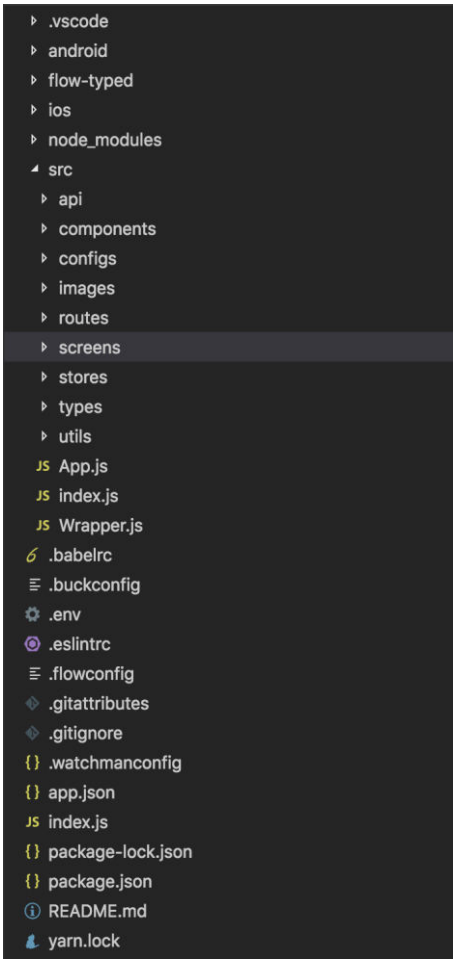

Рисунок 4.7 — Структура директорій у проекті

Розглянемо кожну директорію на предмет використання її модулів у проекті та проаналізуємо структуру проекту в цілому.

Директорія .vscode є опціональною, тут зберігається конфіг, в якому написано правила для написання коду в Microsoft Visual Studio Code. Директорії android та ios містять в собі проекти на відповідних мовах, а це Java та Swift. Саме у них відбувається побудова проекту та його запуск в емуляторі. Директорія flow-typed призначена для приєднання та порівняння типів скачаних модулів, які знаходяться у папці node_modules. Файли, які знаходяться нижче директорій, відповідають за різні правила розгортання додатку, правила коригування коду тощо.

Основною директорією з якою ми працюємо є src. У директорії api ми зберігаємо об’єкти функцій для запитів до API. У components у нас розміщаються спільні модулі, переважно це модулі для UI: поля вводу, поле вибору, кнопки, модальні вікна, таби тощо. У configs зберігаються конфіги для кольорів, текстів, регулярних виразів. Директорія images говорить сама за себе. У директорії utils знаходяться допоміжні функції, які потрібні для роботи зі структурами даних. Директорія types містить типи.

Головними директоріями проекту є routes, screens, stores. Директорія routes (рисунок 4.8) включає в себе модуль маршрутизатора, в ній розміщені шляхи та сторінки, які з’єднанні з цими шляхами.

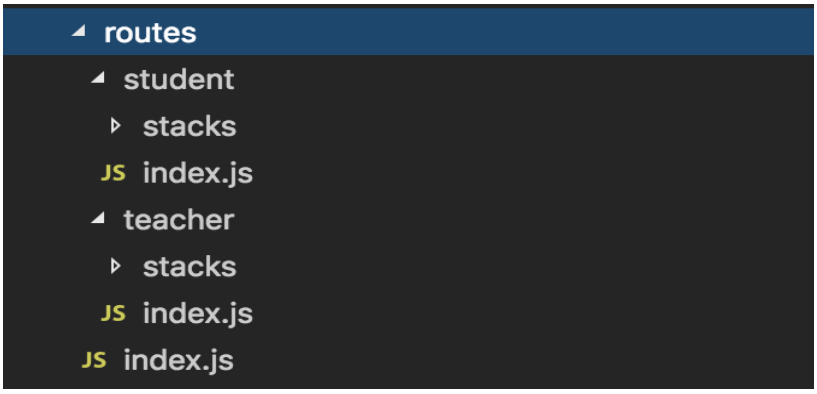

Рисунок 4.8 — Структура директорії “routes”

Усі екрани (скріни) знаходяться у screens (рисунок 4.9), директорія містить спільні екрани та окремі екрани для ролі викладача та студента.

У директорії stores (рисунок 4.10) знаходяться ті самі сховища, які надає бібліотека керування станом MobX. Також є розділення сховищ як для викладача, так і для студента.

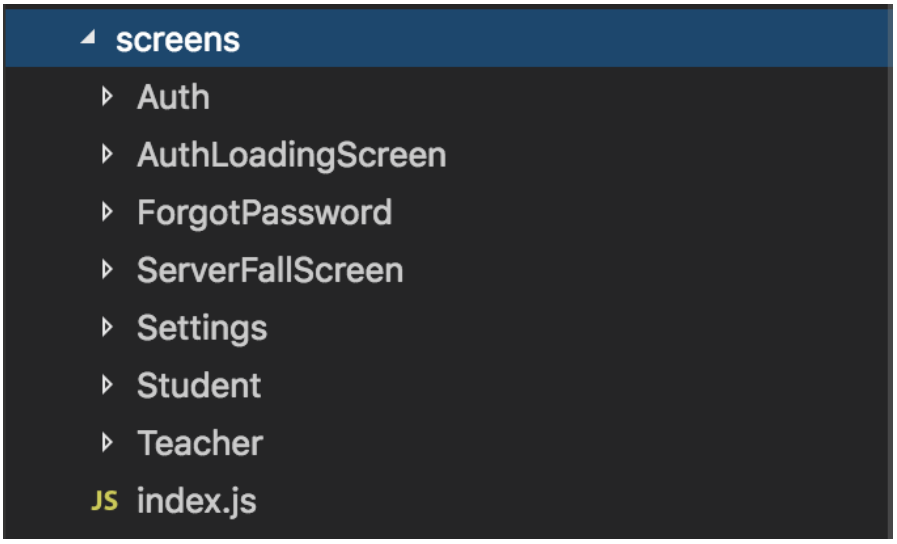

Рисунок 4.9 — Структура директорії “screens”

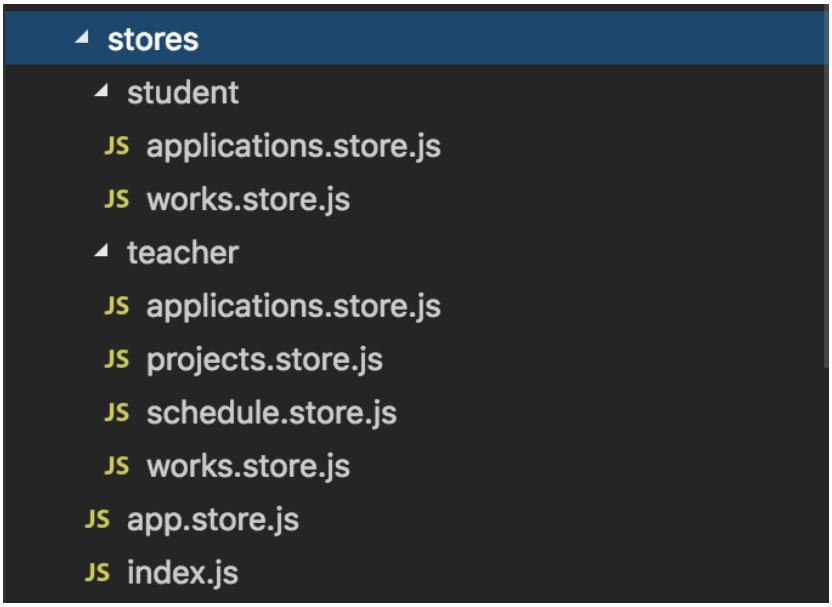

Рисунок 4.10 — Структура директорії “stores”

Таким чином, архітектура проекту побудована вдало і має роздільний характер. Проект складається з незалежних модулів та здатен до масштабованості та додавання нового функціоналу.

Висновки до розділу

У даному розділі був проведений опис програмної реалізації системи управління дипломними проектами. Був проведений аналіз архітектури програмної системи, наведено приклади функціоналу. Пояснено те, як потрібно запускати проекти на React Native. Показано архітектуру мобільного додатку, структуру директорій, пояснено правильність підходу до розробки архітектури.

Перевагою фреймворку React Native є те, що можна самостійно створити структуру, яка буде зручна, немає чітких правил як та що потрібно робити.

Але цей підхід з іншого боку є недоліком, так як недосвідчені розробники можуть побудувати архітектуру неправильно, не за патернами, і в результаті це виходить в складно підтримуваний проект, який потім переписують або довго займаються рефакторингом.

Тому при розробці проекту потрібно враховувати усі аспекти побудови архітектури, адже це дуже важливо.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

5.1 Інсталяція та експлуатаційні характеристики

Мобільний додаток можна встановити через влаштований установник операційної системи Android. Для цього потрібно завантажити на мобільний пристрій, який має операційну систему Android, apk-файл та погодитись на усі умови встановлення даного додатку. Після завантаження додаток буде готовий до роботи.

Також даний додаток можна встановити через магазин мобільних додатків Google Play, мова про якого йшлася у третьому розділі. Назва додатку — ApepsWorks.

Додаток підтримується на усіх мобільних пристроях, які працюють на базі Android починаючи з версії 6.0. Додатковими характеристиками є діагональ екрану, яка починається з 4.5 дюйма, розширення екрану починається з 720x1280 та щільністю від 320 - XHDPI.

5.2 Робота користувача в системі

Як у більшості додатків, робота користувача в системі починається з авторизації користувача (рисунок 5.1). Було розроблено авторизацію користувача з можливістю відновлення паролю (рисунок 5.2).

Відновлення паролю є важливим функціоналом для кожної системи, незалежно від того чи це мобільний додаток чи веб-додаток. Для відновлення пароля необхідно перейти на екран відновлення паролю, натиснувши посилання “Забули пароль?”, після чого ввести свою електронну пошту та роль.

Після виконання даної функції, на введену користувачем пошту прийде пароль, а в додатку з’явиться модальне вікно з повідомленням про успішну операцію.

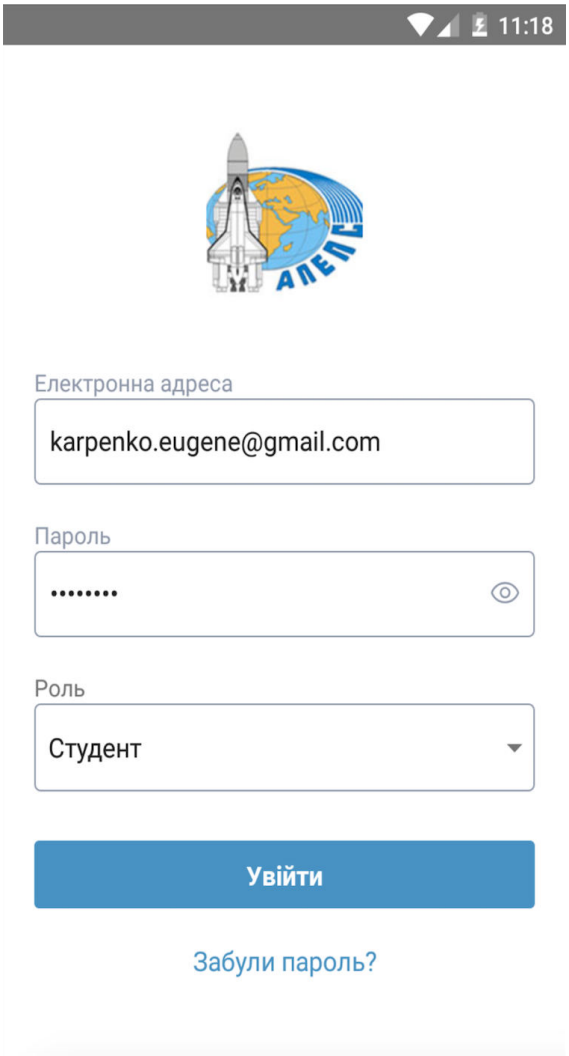

Рисунок 5.1 — Авторизація користувача

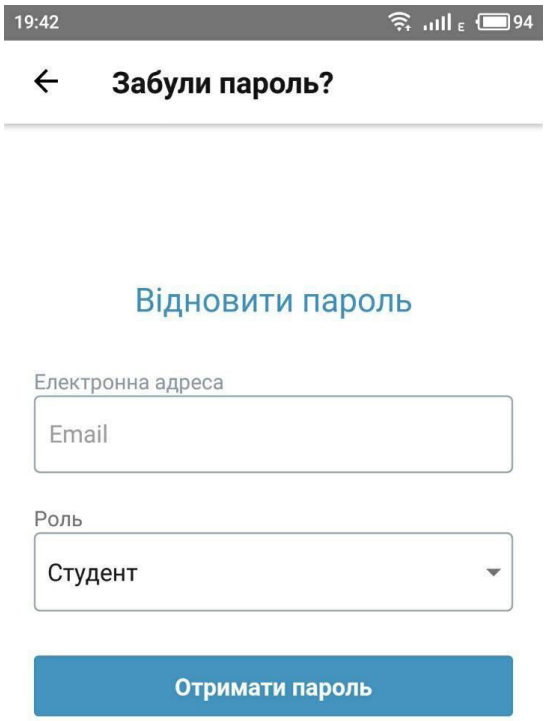

Рисунок 5.2 — Відновлення паролю

Користувачі обох ролей мають спільні екрани, а саме інформація по лабораторіям кафедри та екран профілю. Лабораторії виводяться у вигляди списку, кожен пункт якого має додаткові кнопки для розгортання відповідної інформації.

На вкладці “Лабораторії” (рисунок 5.3) відображено список лабораторій. Також надана можливість перегляду напрямів кожної з лабораторій, їхню детальну інформацію.

До того ж на даному екрані можна побачити список викладачів (рисунок 5.4), їхнє ім’я, звання та посаду.

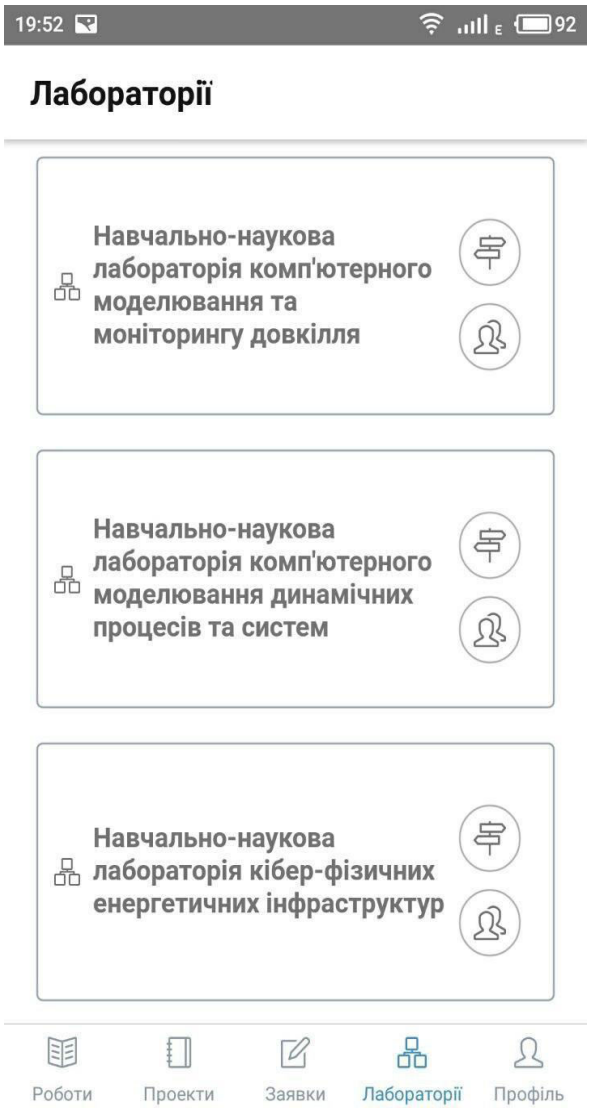

Рисунок 5.3 — Лабораторії

Рисунок 5.4 — Викладачі

Екран профілю (рисунок 5.5) містить в собі персональну інформацію користувача, його фото, електронну пошту, телефон тощо.

Існує можливість редагування профілю (рисунок 5.6), а саме заміни фото, натиснувши на посилання “Завантажити фото”, редагування телефону та електронної пошти.

У профілі викладача відображена додаткова інформація по навантаженню, у профілі студента ця інформація відсутня. Також у профілі є кнопка для виходу з системи.

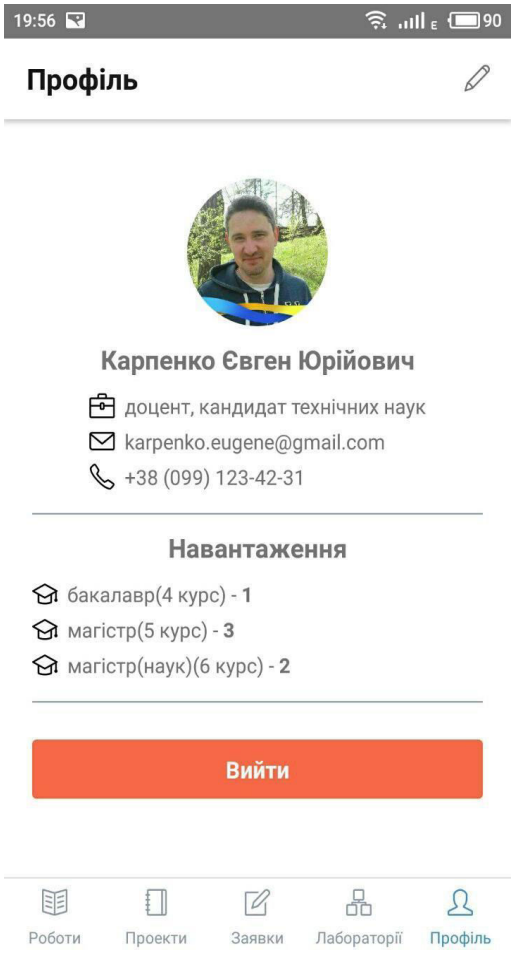

Рисунок 5.5 — Профіль користувача

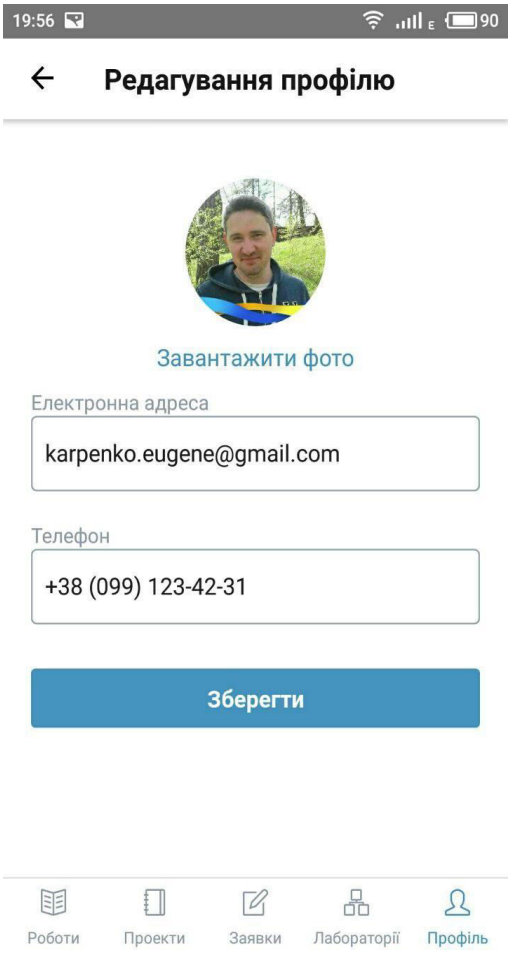

Рисунок 5.6 — Редагування профілю

Це були усі спільні екрани даної системи. Перейдемо до окремих екранів обох ролей: викладача і студента.

5.2.1. Екрани для ролі “Викладач”

На вкладці “Дипломні роботи” (рисунок 5.7) у викладача відображено список дипломних робіт. Існує два типи дипломних робіт — “вільні” та “зайняті”. Якщо студент закріплений за темою дипломної роботи, то дана тема є зайнятою.

Також викладач може створити, редагувати, видаляти тему (рисунок 5.8), а також існує функція знаття студента з теми, після чого тема стає вільною і знову доступна для вибору студентів.

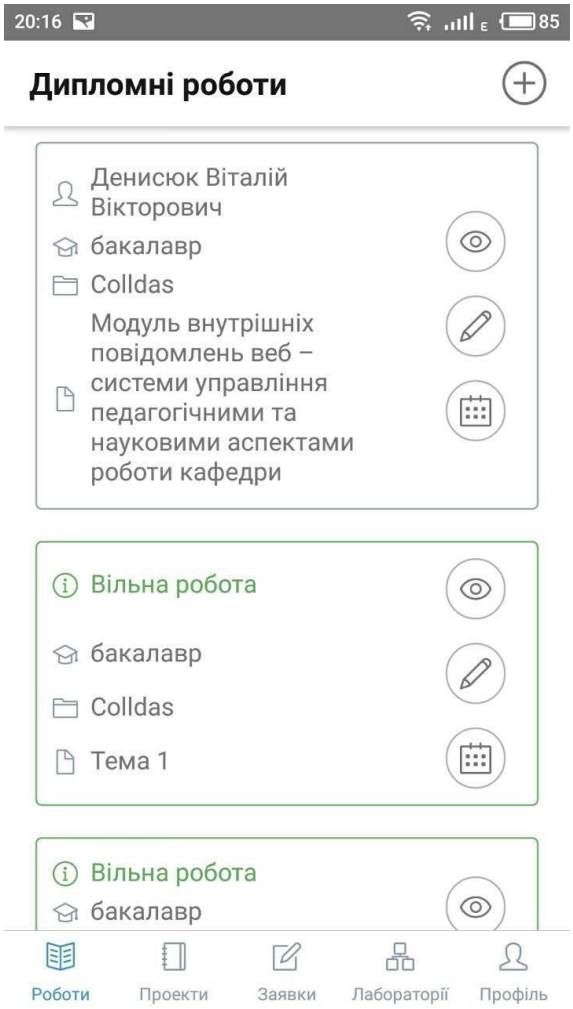

Рисунок 5.7 — Вкладка “Дипломні роботи”

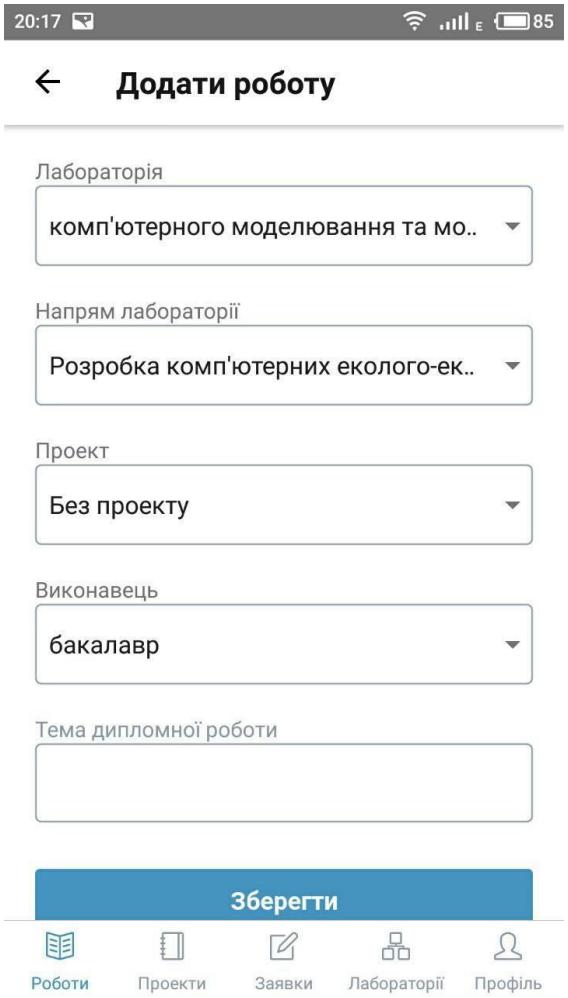

Рисунок 5.8 — Вкладка “Додати роботу”

При кліку на кнопку з зображенням календаря переходимо на екран графіку виконання дипломної роботи. Подібна функція існує у студента, яка відображена у додатку як окрема вкладка.

Графік виконання представляє собою список завдань (рисунок 5.9), які потрібно виконати за певний період часу. У кожному завданні є список документів, які потрібно зробити студенту та здати або просто підготувати ці документи.

Для викладача існує функція виставлення статусу, серед яких є такі: “Виконано” , “На виконанні”, “Не виконано”. Також є функція написання приміток для кожного завдання (рисунок 5.10).

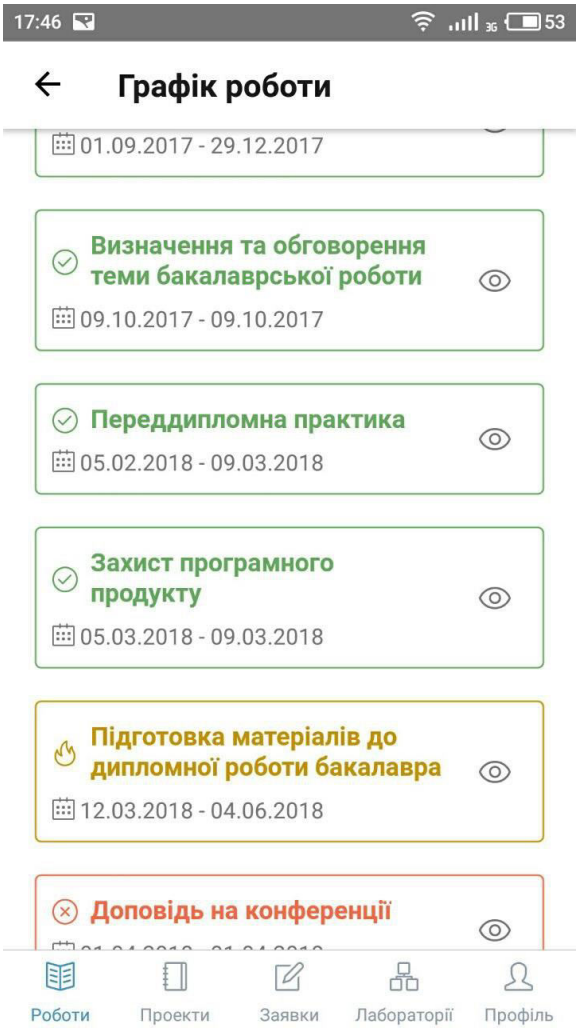

Рисунок 5.9 — Екран “Графік роботи”

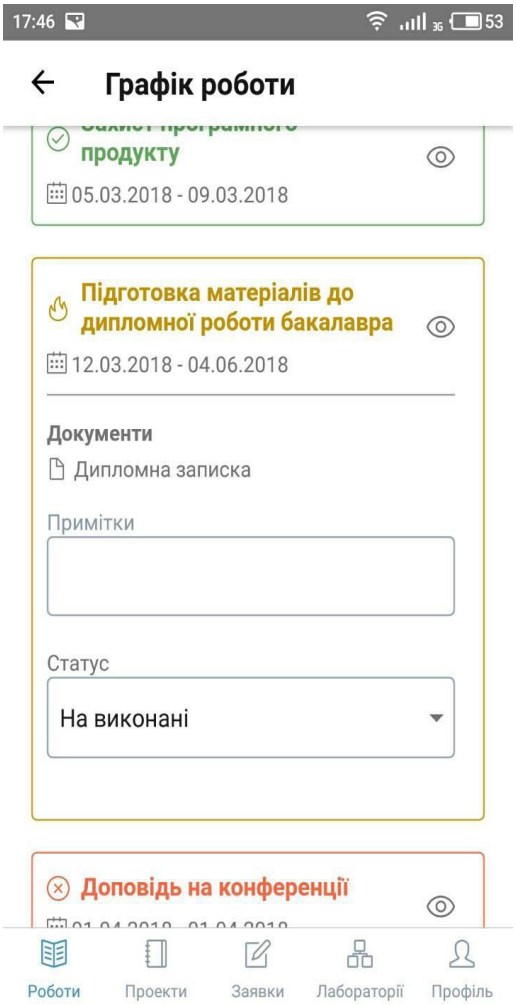

Рисунок 5.10 — Виставлення статусу

Дипломна робота може відноситись до проекту (рисунок 5.11). Проекти по аналогії можна переглядати, створювати (рисунок 5.12), редагувати та видаляти (рисунок 5.13). Це все можна робити знаходячись на вкладці “Проекти”.

Проект — це одна велика робота, яка розділена на модулі. Зазвичай, над проектом працюють кілька студентів і кожен має описати та зробити свій модуль, свою частину проекту.

З цього випливає, що студенти працюють над одним і тим же функціоналом, але кожен модуль має свої відмінності і студенти реалізовують їх якнайкраще.

Тому даний екран є важливим функціоналом для викладача.

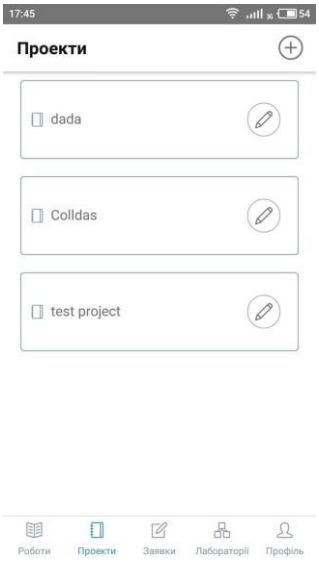

Рисунок 5.11 — Вкладка “Проекти”

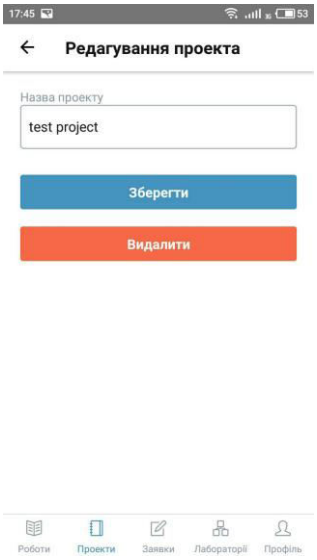

Рисунок 5.12 — Редагування проекту

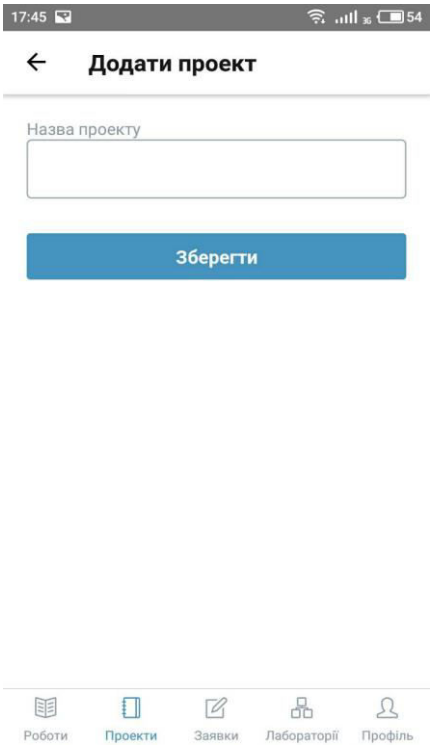

Рисунок 5.13 — Вкладка “Проекти”

У вкладці “Заявки” (рисунок 5.14) викладач може переглядати надіслані заявки від студентів. На даному екрані є можливість перегляду заявок як від бакалаврів, так і від магістрів.

Після прийняття заявки студента у викладача стане на одну вільну тему дипломної роботи менше, оскільки після прийняття заявки студент автоматично закріплюється за обраною темою дипломної роботи.

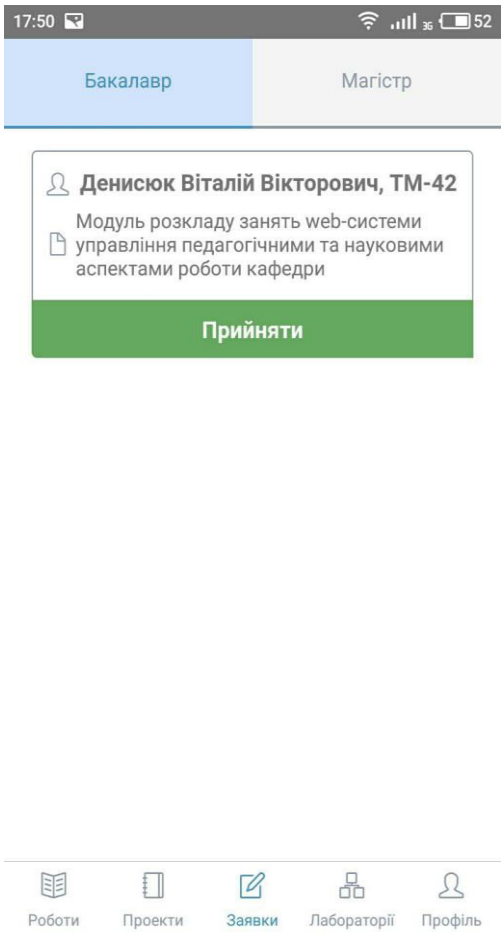

Рисунок 5.14 — Вкладка “Заявки”

Це весь функціонал для ролі “Викладач”. Перейдемо до можливостей ролі “Студент”.

5.2.2. Екрани для ролі “Студент”

Якщо студент ще не обрав тему дипломної роботи, йому пропонується обрати тему серед багатьох запропонованих. У кожній темі можна подивитись за якою лабораторією та напрямом закріплена дана робота, натиснувши на кнопку зі зображенням ока.

При кліку на кнопку із зображенням галочки студент подає заявку на тему, яку обрав в списку тем (рисунок 5.15).

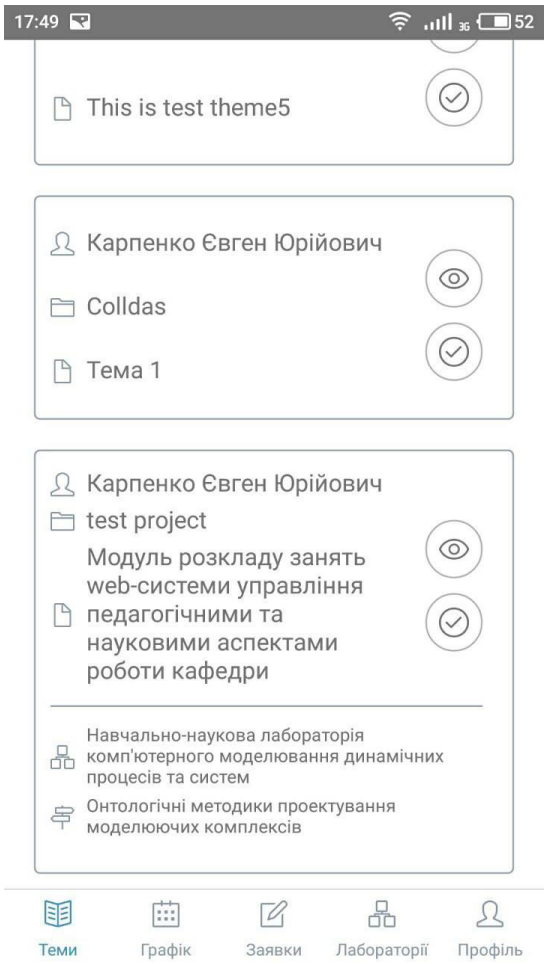

Рисунок 5.15— Вкладка “Теми”

Якщо ж студент закріплений за темою, яку вже обрав, тоді студенту замість вкладки “Теми” буде надано вкладку “Моя тема” (рисунок 5.16).

На даному екрані студенту надається інформація про його дипломного керівника, інформація про лабораторію, напрям та назва теми.

Оскільки студент закріплений за темою, тому йому буде доступно графік виконання дипломної роботи (рисунок 5.17).

Саме там він зможе побачити усі завдання, які потрібно йому зробити. Зможе отримувати примітки від дипломного керівника.

Внаслідок такого функціоналу студент зможе вчасно та якісно виконати дипломну роботу, адже вся інформація доступна онлайн в смартфоні, що дозволяє зекономити час на комунікацією з викладачем.

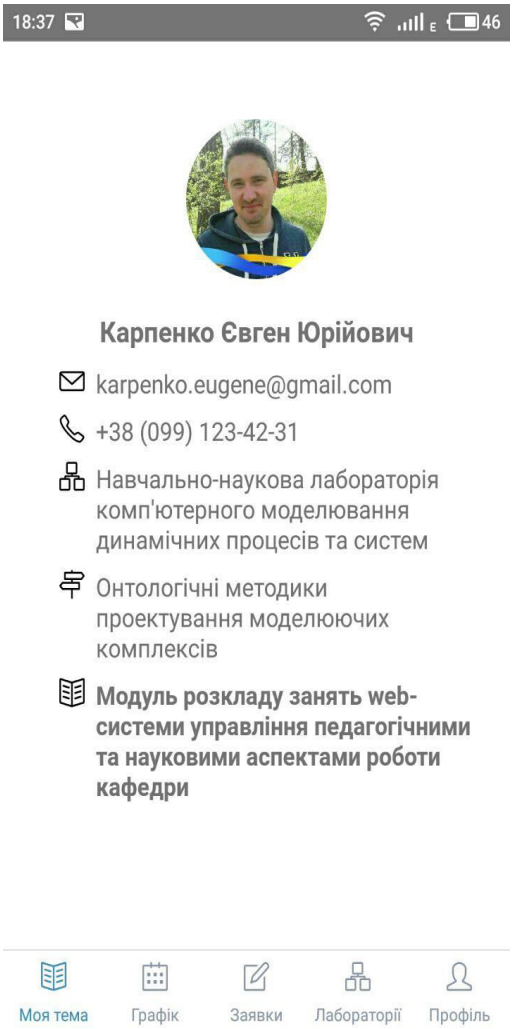

Рисунок 5.16 — Вкладка “Моя тема”

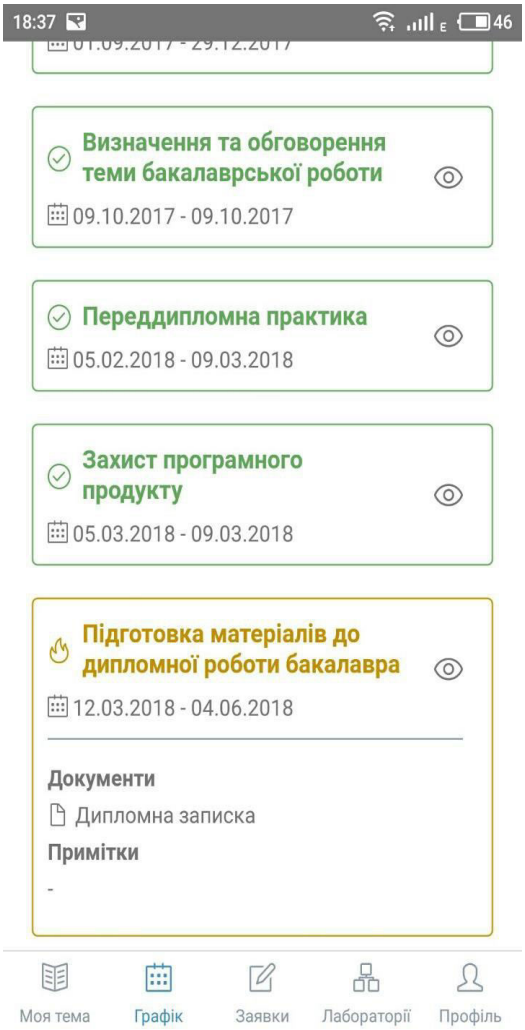

Рисунок 5.17 — Вкладка “Графік”

У вкладці “Заявки” студент може переглядати подані ним заявки на різні теми дипломних робіт, також є можливість відмінити заявку.

Висновки до розділу

У даному розділі було проведено опис методики роботи користувача з програмною системою. Мобільний додаток має функціонал авторизації. Разом з авторизацією важливу роботу виконує функціонал відновлення паролю.

Система має дві ролі — “Викладач” та “Студент”. Як у викладача, так і у студента є спільний функціонал, а саме профіль та перегляд інформації по лабораторіям.

Окремо у кожної ролі є свої можливості керування дипломною роботою, а саме подача та прийняття заявок, перегляд і редагування графіку виконання дипломної роботи тощо.

ВИСНОВКИ

Головною метою дипломного проекту було створити мобільний додаток системи управління проектами.

Даний дипломний проект був написаний на мові JavaScript за допомогою фреймворку React Native, що включає в себе принципи бібліотеки React.

Вивчено та застосовано основні деталі та засоби розробки для мобільних додатків, зокрема — емулятори операційної системи Android.

Проаналізовано існуючі системи управління проектами, визначено їх переваги та недоліки. Передбачені рішення проблем управління дипломними проектами та можливі вдосконалення для якіснішої роботи над ними.

З використанням мови JavaScript та фреймворку React Native реалізовано мобільний додаток, який надає можливість студентам та викладачам керувати дипломною роботою, звіряти терміни, подавати та приймати заявки, переглядати необхідну та корисну інформацію.

Мобільний додаток успішно пройшов тестування на кількох пристроях з операційною системою Android починаючи з версії 6.0.

Реалізовані можливості програмного продукту повністю задовольняють поставлені задачі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Флэнаган Д. JavaScript. Подробное руководство, 6е издание. – Пер. с англ. – СПб: СимволПлюс, 2012. – 228-241 с.
2. Пауэрс Ш. Изучаем Node.js — СПб.: Питер, 2014. — 123-136 с.
3. Bonnie Eisenman. Learning React Native: Building Native Mobile Apps with JavaScript, 2nd edition — Published by O'Reilly Media, 2016 — P. 150-400
4. Ethan Holmes, Tom Bray. Getting Started with React Native — Published by Packt Publishing Ltd, 2015 — P. 129-172
5. Алекс Бэнкс. React и Redux. Функциональная веб-разработка — СПб.: Питер, 2019. — 220-240 с.
6. Стоян Стефанов. JavaScript. Шаблоны — СПб.: Символ-Плюс, 2011. – 272 с
7. Pavan Podila and Michel Weststrate. MobX Quick Start Guide — Published by Packt Publishing Ltd, 2018 — P. 10-112
8. Bill Phillips and Brian Hardy. Android Programming: The Big Nerd Ranch Guide, 1st edition — Published by Big Nerd Ranch Inc., 2013 — P. 221-312
9. MySQL. Руководство администратора MySQL. Administrator's Guide. — М.: Издательский дом «Вильямс», 2005. — С. 624.
10. Итан Браун. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript = Web Development with Node and Express / Итан Браун. — Санкт-Петербург: Питер, 2017. — 336 с.
11. Mateusz Grzesiukiewicz. Hands-On Design Patterns with React Native — Published by Packt Publishing Ltd, 2018 — P. 59-90
12. Frank Zammetti. Practical React Native — Published by Apress, 2018 — P. 96-133
13. Stoyan Stefanov. React: Up & Running: Building Web Applications — Published by O'Reilly Media, 2016 — P. 222
14. Eric Elliott. Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Modern JS Libraries — Published by O'Reilly Media, 2016 — P. 180-250

15. Kyle Simpson. You Don't Know JS: ES6 & Beyond — Published by O'Reilly Media, 2015 — P. 278
16. Kyle Simpson. You Don't Know JS: Async & Performance — Published by O'Reilly Media, 2015 — P. 150-223
17. Shelley Powers. Learning JavaScript, 2nd Edition — Published by O'Reilly Media, 2008 — P. 120-154
18. Современный учебник JavaScript [Електронний ресурс] – Режим доступу до ресурсу: https://learn.javascript.ru/.
19. Microsoft Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/.
20. React — A JavaScript library for building user interfaces [Електронний ресурс] — Режим доступу до ресурсу: https://reactjs.org/.
21. React Native — A framework for building native apps using React [Електронний ресурс] — Режим доступу до ресурсу: https://facebook.github.io/react-native/.
22. MobX Documentation [Електронний ресурс] — Режим доступу до ресурсу: https://mobx.js.org/.
23. Genymotion 2.12 User Guide [Електронний ресурс] — Режим доступу до ресурсу: https://docs.genymotion.com/latest/pdf/PDF_User_Guide/Genymotion-2.12-User-Guide.pdf
24. Jira Software Features [Електронний ресурс] — Режим доступу до ресурсу: https://ru.atlassian.com/software/jira/features
25. Trello [Електронний ресурс] – Режим доступу до ресурсу: https://trello.com/home

Додаток А

Мобільний додаток системи управління дипломними проектами

Специфікація

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б

Аркушів 1

Київ — 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б	Пояснювальна записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б 12-1	App.js, Wrapper.js, index.js	Основні компоненти, що включають в себе інші
УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б 13-1	Додаток В.docx	Опис програмного модуля

Додаток Б

Мобільний додаток системи управління дипломними проектами

Текст програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б 12-1

Аркушів 14

Київ — 2019

```
// Головний модуль index.js, що реєструє та запускає додаток

/** @format */

import { AppRegistry } from 'react-native';
import 'es6-symbol/implement';
import App from '@src/App';
import { name as appName } from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

```
// Головний модуль App.js, включає в себе компонент Wrapper, до якого підключені всі інші модулі. Підключено провайдер для сховищ за допомогою бібліотеки mobx-react.

/* @flow */
/* REACT */
import React, { Component } from 'react';

/* MODULES */
import { Provider } from 'mobx-react';
import SplashScreen from 'react-native-splash-screen';

/* CUSTOM MODULES */
import Wrapper from './Wrapper';
import allStores from '@stores';

/* TYPES */
type _t_props = {};
type _t_state = {};

export default class App extends Component<_t_props, _t_state> {

 state = {}

 componentDidMount() {
   window.setTimeout(() => {
     SplashScreen.hide();
   }, 1000);
 }

 render() {
   return (
     <Provider {...allStores}>
       <Wrapper />
     </Provider>
   );
```

```
 }
}


// Головний компонент Wrapper.js, в якому підключено контейнер з шляхами до екранів додатку,
використовується сервіс для навігації NavigationService

/* @flow */
/* REACT */
import React, { Component } from 'react';

/* MODULES */
import { inject, observer } from 'mobx-react';

/* CUSTOM MODULES */
import AppContainer from './routes';
import { NavigationService } from '@utils';

/* MODULES */

/* TYPES */
type _t_props = {|
 user: Object
|};
type _t_state = {};

@inject(({ appStore }) => ({
 user: appStore.user,
}))
@observer
export default class Wrapper extends Component<_t_props, _t_state> {

 state = {}

 render() {
   const { user } = this.props;

   return (
     <AppContainer
       ref={(navigatorRef) => {
         NavigationService.setTopLevelNavigator(navigatorRef);
       }}
       screenProps={{
         handler: () => {},
         studentWithTheme: !!user.thesisId
       }}
     />
```

```
    );
  }
}


// Модуль у якому зібрано усі шляхи додатку за допомогою бібліотеки react-navigation,
підключено модулі скрінів.

/* @flow */
import {
 createSwitchNavigator,
 createStackNavigator,
 createAppContainer
} from 'react-navigation';

import {
 Auth, ForgotPassword,
 AuthLoadingScreen,
 ServerFallScreen
} from '@src/screens';

import TeacherScreens from './teacher';
import StudentScreens from './student';

const AuthStack = createStackNavigator({
 SignIn: Auth, ForgotPassword
});

const AppNavigator = createSwitchNavigator(
 {
   AuthLoading: AuthLoadingScreen,
   ServerFall: ServerFallScreen,
   Auth: AuthStack,
   Teacher: TeacherScreens,
   Student: StudentScreens
 },
 {
   initialRouteName: 'AuthLoading'
 }
);

const AppContainer = createAppContainer(AppNavigator);

export default AppContainer;


// Модуль шляхів ролі “Викладач”, для кожного шляху описані свої характеристики і скріни, які
```

відносяться до цих шляхів.

```
/* @flow */
/* REACT */
import React from 'react';

/* MODULES */
import {
 createStackNavigator,
 createBottomTabNavigator,
} from 'react-navigation';
import Icon from 'react-native-vector-icons/SimpleLineIcons';

/* CUSTOM MODULES */
import { Applications } from '@src/screens/Teacher';
import { COLOR_CONFIG } from '@configs';
import { WorksStack, LabsStack, ProfileStack, ProjectsStack } from './stacks';

const TeacherTabNavigator = createBottomTabNavigator(
 {
   Works: {
     screen: WorksStack,
     navigationOptions: {
       tabBarLabel: 'Роботи',
       tabBarIcon: ({ tintColor }: Object) => (
         <Icon name="book-open" color={tintColor} size={18} />
       )
     }
   },
   Projects: {
     screen: ProjectsStack,
     navigationOptions: {
       tabBarLabel: 'Проекти',
       tabBarIcon: ({ tintColor }: Object) => (
         <Icon name="notebook" color={tintColor} size={18} />
       )
     }
   },
   Applications: {
     screen: Applications,
     navigationOptions: {
       tabBarLabel: 'Заявки',
       tabBarIcon: ({ tintColor }: Object) => (
         <Icon name="note" color={tintColor} size={18} />
       )
     }
   },
```

```
    LabsStack: {
      screen: LabsStack,
      navigationOptions: {
        tabBarLabel: 'Лабораторії',
        tabBarIcon: ({ tintColor }: Object) => (
          <Icon name="organization" color={tintColor} size={18} />
        )
      }
    },
    ProfileStack: {
      screen: ProfileStack,
      navigationOptions: {
        tabBarLabel: 'Профіль',
        tabBarIcon: ({ tintColor }: Object) => (
          <Icon name="user" color={tintColor} size={18} />
        )
      }
    },
  },
  {
    initialRouteName: 'Works',
    navigationOptions: ({ navigation }) => {
      const { routeName } = navigation.state.routes[navigation.state.index];

      return {
        header: null,
        headerTitle: routeName
      };
    },
    tabBarOptions: {
      activeTintColor: COLOR_CONFIG.BLUE,
      inactiveTintColor: COLOR_CONFIG.GRAY,
    },
  }
);

const TeacherStackNavigator = createStackNavigator(
  {
    TeacherTabNavigator
  }
);

export default TeacherStackNavigator;
```

// Модуль екрану авторизації, який містить форму авторизації, яка знаходиться в окремому файлі, підключена функція login, яка взята зі сховища, що виконує запит на сервер.

```
// @flow

/* REACT */
import React, { Component } from 'react';

/* REACT NATIVE */
import { ScrollView } from 'react-native';

/* MODULES */
import { inject, observer } from 'mobx-react';

/* TYPES */
import type { request } from '@src/types';

/* CUSTOM MODULES */
import LoginForm from './LoginForm';

type _t_props = {|
 login: request,
 navigation: Object,
|}

@inject(({ appStore }) => ({
 login: appStore.login,
}))
@observer
export class Auth extends Component<_t_props> {

 static navigationOptions = () => ({ header: null })

 render() {
   const { navigation, login } = this.props;

   return (
     <ScrollView>
       <LoginForm
         login={login}
         navigation={navigation}
       />
     </ScrollView>
   );
 }
}

export default Auth;
```

```
// Модуль форми авторизації, в якій підключено необхідні елементи вводу: Input, Button,
Select, Link. Форма має методи збору інформації та відправлення запиту на сервер.

// @flow

/* REACT */
import React, { Component } from 'react';

/* REACT NATIVE */
import {
 ScrollView, View, Text, AsyncStorage, StyleSheet, Image
} from 'react-native';

/* MODULES */
import { inject, observer } from 'mobx-react';

/* CUSTOM MODULES */
import {
 Button, Input, Select, Link
} from '@components/UI';
import { COLOR_CONFIG, PATTERN_CONFIG, ERRORS_CONFIG } from '@configs';
import { removeField, checkRequired, checkPattern } from '@utils';
import { LOGO } from '@src/images';

/* TYPES */
import type { request } from '@src/types';

type _t_props = {|
 login: request,
 navigation: Object
|}

type _t_state = {|
 form: Object,
 errors: Object,
 errorLogin: boolean
|}

// оголошуємо обов’язкові поля та підключаємо регулярні вирази для перевірки правильності
вводу інформації

const REQUIRED_FIELDS = ['username', 'password', 'role'];
const PATTERN_FIELDS = {
 username: ERRORS_CONFIG.email,
 password: {
   pattern: PATTERN_CONFIG.password,
```

```
   message: 'Довжина пароля не менше 6 символів!'
 }
};

// Ініціалізація полів форми
const INITIAL_FORM = {
 username: '', // 'karpenko.eugene@gmail.com',
 password: '', // 'QluAGE3i',
 role: 'student',
};

@inject(({ appStore }) => ({
 login: appStore.login,
}))
@observer
class LoginForm extends Component<_t_props, _t_state> {

 // стан компонента форми
 state = {
   form: INITIAL_FORM,
   errors: {},
   errorLogin: false
 }

 // Метод життєвого циклу, який виконується одразу при рендері екрану. Тут відбувається
перевірка на те, чи є облікові дані користувача, які користувач вводив раніше
 async componentDidMount() {
   const email = await AsyncStorage.getItem('Email');
   const password = await AsyncStorage.getItem('Password');
   const role = await AsyncStorage.getItem('SaveRole');

   if (email && password && role) {
     this.setState(() => ({
       form: {
         username: email,
         password,
         role
       }
     }));
   }
 }

 clearForm = () => this.setState(() => ({ form: INITIAL_FORM, errors: {} }))

 onChangeText = (key: string, text: string) => {
   this.setState(prevState => ({
     form: { ...prevState.form, [key]: text },
```

```
      errors: removeField(prevState.errors, key),
      errorLogin: false
    }));
  }

  // зберігання токену та ролі користувача у сховищі AsyncStorage
  saveToken = async (token: string, role: string) => {
    try {
      await AsyncStorage.setItem('Auth', token);
      await AsyncStorage.setItem('Role', role);
    } catch (error) {
      console.log(error);
    }
  };

  // зберігання облікових даних користувача у сховищі AsyncStorage
  saveUser = async (email: string, password: string, role: string) => {
    try {
      await AsyncStorage.setItem('Email', email);
      await AsyncStorage.setItem('Password', password);
      await AsyncStorage.setItem('SaveRole', role);
    } catch (error) {
      console.log(error);
    }
  };
```

// метод викликається при кліку на кнопку, відбувається валідація полів, після цього викрнується збір даних та запит на сервер. Якщо отримуємо позитивниу відповідь - зберігаємо токен, роль і облікові дані користувача. Перенаправляємо користувача на екрани викладача або студента в залежності від ролі користувача.

```
  submitHandler = () => {
    if (!checkRequired(this, REQUIRED_FIELDS) && !checkPattern(this, PATTERN_FIELDS)) {
      const { login, navigation } = this.props;
      const { form } = this.state;

      login(
        {
          email: form.username.trim(),
          password: form.password.trim(),
          role: form.role,
        },
        (res: Object) => {
          this.clearForm();
          this.saveUser(form.username.trim(), form.password.trim(), form.role);
          this.saveToken(res.token, res.role);
          navigation.navigate(res.role === 'lector' ? 'Teacher' : 'Student');
```

```
      },
      () => this.setState(() => ({ errorLogin: true })),
    );
  }
}

render() {
  const { navigation } = this.props;
  const {
    form: { username, password, role }, errors, errorLogin
  } = this.state;

  return (
    <ScrollView contentContainerStyle={styles.container}>
      <View>
        <View
          style={{
            marginBottom: 40,
            alignItems: 'center'
          }}
        >
          <Image
            source={LOGO}
            style={{ width: 100, height: 100 }}
          />
          {/* <Text style={{ color: COLOR_CONFIG.BLUE, fontSize: 26 }}> - APEPS - </Text> */}
        </View>
        <Input
          label="Електронна адреса"
          value={username}
          isError={!!errors.username}
          errorText={errors.username}
          onChangeText={text => this.onChangeText('username', text)}
        />
        <Input
          label="Пароль"
          value={password}
          isError={!!errors.password}
          errorText={errors.password}
          isPasswordField
          onChangeText={text => this.onChangeText('password', text)}
        />
        <Select
          label="Роль"
          value={role}
          options={[
            { label: 'Викладач', value: 'lector' },
```

```
            { label: 'Студент', value: 'student' },
          ]}
          onValueChange={text => this.onChangeText('role', text)}
        />
        {
          errorLogin && (
            <Text style={{ textAlign: 'center', color: COLOR_CONFIG.RED }}>Помилка! Дані
некоректні!</Text>
          )
        }
        <Button
          label="Увійти"
          color="blue"
          onPress={this.submitHandler}
        />
        <View style={{ alignItems: 'center', marginTop: 10 }}>
          <Link
            label="Забули пароль?"
            onPress={() => navigation.navigate('ForgotPassword')}
          />
        </View>
      </View>
    </ScrollView>
  );
 }
}
// Стилі
const styles = StyleSheet.create({
 container: {
   flex: 1,
   flexDirection: 'column',
   justifyContent: 'center',
   alignContent: 'center',
   height: 600,
   padding: 20,
 },
});

export default LoginForm;



// Модуль сховища app.store.js - є прикладом сховища.

// @flow

/* REACT */
```

```
/* REACT NATIVE */
import { AsyncStorage } from 'react-native';

/* MODULES */
import { observable, action } from 'mobx';

/* CUSTOM MODULES */
import { ApiCaller, NavigationService } from '@utils';
import { UserRequest, ReferenceRequest } from '@api';
import projectsStore from './teacher/projects.store';

class AppStore {

 // поля, які зберігають інформацію

 @observable user = {};

 @observable labs = [];

 @observable directions = [];

 @observable degrees = [];

 @observable directionInfo = [];

 @observable loading = false;

 // методи, які виконують запит на сервер, використовуючи конфіг даних про запит, та метод
this.request, який обробляє інформацію для запиту

 @action
 login = async (payload: Object, onSuccess: Function, onFail: Function) => {
   await this.request(this, {
     action: UserRequest.login,
     payload,
     onSuccess: (res) => {
       if (onSuccess) { onSuccess(res); }
       this.user = {
         ...res.user,
         role: res.role,
         token: res.token
       };
       this.getReferenceInfo();
       projectsStore.getProjects();
     },
     onFail: (res) => {
       if (onFail) { onFail(res); }
```

```
      },
      loading: 'loading'
    });
  }

  // метод, який збирає інформацію для запиту та виконує його, використовуючи сервіс ApiCaller,
який був описаний раніше
  @action
  request = async (ctx: Object, config: Object) => {
    if (config.loading) { ctx[config.loading] = true; }

    try {
      const {
        action: { path, method, requiredAuth },
        payload, onSuccess, onFail, loading
      } = config;
      let headers = null;
      if (requiredAuth) {
        const token = await AsyncStorage.getItem('Auth');
        const role = await AsyncStorage.getItem('Role');
        headers = { token, role };
      }
      const url = path(payload);

      const { data } = await ApiCaller(url, method, payload || null, headers);
      if (data.code && data.code >= 400) {
        onFail(data);
      } else {
        onSuccess(data);
      }
      if (loading) { ctx[loading] = false; }
    } catch (e) {
      config.onFail(e);
      if (config.loading) { ctx[config.loading] = false; }
      NavigationService.navigate('ServerFall', { error: e });
    }
  }
}

const appStore = new AppStore();

export default appStore;

// Файл-конфіг package.json, який містить правила про проект та встановлені модулі з npm

{
  "name": "apeps-mobile-app",
```

```
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "node node_modules/react-native/local-cli/cli.js start",
    "test": "jest"
  },
  "dependencies": {
    "axios": "^0.18.0",
    "es6-symbol": "^3.1.1",
    "mobx": "4.8.0",
    "mobx-react": "^5.4.3",
    "moment": "^2.24.0",
    "react": "16.6.3",
    "react-native": "0.57.8",
    "react-native-config": "^0.11.7",
    "react-native-gesture-handler": "^1.0.12",
    "react-native-image-crop-picker": "^0.22.0",
    "react-native-keyboard-aware-scroll-view": "^0.8.0",
    "react-native-splash-screen": "^3.2.0",
    "react-native-text-input-mask": "^1.0.1",
    "react-native-vector-icons": "^6.2.0",
    "react-navigation": "^3.0.9",
    "rn-fetch-blob": "^0.10.15"
  },
  "devDependencies": {
    "@babel/plugin-proposal-decorators": "^7.3.0",
    "babel-eslint": "^10.0.1",
    "babel-jest": "23.6.0",
    "babel-plugin-module-resolver": "3.1.1",
    "eslint": "^5.12.1",
    "eslint-config-airbnb": "^17.1.0",
    "eslint-plugin-flowtype": "^3.2.1",
    "eslint-plugin-import": "^2.14.0",
    "eslint-plugin-jsx-a11y": "^6.1.2",
    "eslint-plugin-module-resolver": "^0.8.0",
    "eslint-plugin-react": "^7.12.4",
    "eslint-plugin-react-native": "^3.6.0",
    "flow-bin": "0.81.0",
    "jest": "23.6.0",
    "json5": "^2.1.0",
    "metro-react-native-babel-preset": "0.51.1",
    "react-test-renderer": "16.6.3"
  },
  "jest": {
    "preset": "react-native"
  }
}
```

Додаток В

Мобільний додаток системи управління дипломними проектами

Опис програми

УКР.НТУУ”КПІ”ІМ.ІГОРЯ_СІКОРСЬКОГО_ТЕФ_АПЕПС_ТВ51146_19Б 13-1

Аркушів 8

Київ — 2019

АНОТАЦІЯ

Додаток являє собою сервіс управління дипломними проектами.

Додаток має дві ролі — викладач та студент. У функціонал викладача входить: створення, редагування, видалення тем, прийняття заявок, перегляд та редагування графіку виконання дипломної роботи. Студент може переглядати теми, подавати заявки на кілька тем, переглядати графік виконання дипломної роботи.

У кожної ролі є спільний функціонал такий як профіль та екран з лабораторіями та інформацією про них.

Модуль було розроблено з використанням мови програмування JavaScript, фреймворку React Native, бібліотеки керуванням стану програми MobX, емулятора Genymotion, середовища розробки Visual Studio Code.

ЗМІСТ

1. ЗАГАЛЬНІ ВІДОМОСТІ 77
2. ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ 78
3. ОПИС ЛОГІЧНОЇ СТРУКТУРИ 79
4. ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУЮТЬСЯ 80
5. ВИКЛИК І ЗАВАНТАЖЕННЯ 81
6. ВХІДНІ ТА ВИХІДНІ ДАНІ 82

ЗАГАЛЬНІ ВІДОМОСТІ

Для функціонування програми потрібно встановити Node.js. Разом з ним завантажується менеджер пакетів npm. Після цього потрібно встановити react-native-cli. Для цього виконуємо команди “npm install -g react-native-cli”.

Також перед запуском проекту потрібно мати встановлений Java Development Kit 8. Для запуску додатку потрібно відкрити проект у Visual Studio Code або відкрити термінал з директорії проекту і виконати наступні команди: “npm install && react-native run-android”. Перед цим потрібно завантажити емулятор Genymotion та запустити емулятор мобільного пристрою.

Основний модуль був написаний на мові JavaScipt за допомогою фреймворку React Native у середовищі розробки Visual Studio Code.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Мобільний додаток складається з двох модулів: модуль “Викладач” та модуль “Студент”. Дані модулі були розроблені для вирішення проблеми керування дипломними роботами. Проблему вирішено завдяки мові JavaScript та фреймворку React Native.

Модуль “Студент” має такі функції:

— пошук, перегляд та вибір теми дипломної роботи;
— подача заявки на кілька дипломних робіт;
— відміна поданих заявок;
— перегляд інформації по поточній дипломній роботі після підтвердження заявки викладачем;
— перегляд графіку виконання дипломної роботи;
— перегляд інформації по лабораторіям кафедри;
— перегляд та можливість редагування профілю.

Модуль “Викладач” має такі функції:

— перегляд, створення, редагування та видалення теми дипломної роботи;
— перегляд, створення, редагування та видалення проектів;
— прийняття заявок від студентів(бакалавр, магістр);
— зняття студента з теми дипломної роботи;
— перегляд графіку виконання дипломної роботи з можливістю редагування статусу кожного етапу та внесення певних вказівок, приміток;
— перегляд інформації по лабораторіям кафедри;
— перегляд та можливість редагування профілю.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Логічна структура програми складається з методів та функції, які виконують поставлені задачі. Програма містить директорію компонентів, де знаходяться спільні компоненти для користувацького інтерфейсу. Одним з головних модулів є директорія, яка містить екрани. За маршрутизацію додатку відповідають модулі в директорії шляхів. За збереження даних та їх зміну відповідають модулі сховищ, які знаходяться в директорії сховищ. Також існують допоміжні модулі, які формують структуру програми — це конфіги, допоміжні функції тощо.

Усі модулі програми є незалежними та здатні до масштабування. Саме завдяки правильній побудові структури програми, вона є гнучкою, легкою в подальшій розробці та підтримці.

ТЕХНІЧНІ ЗАСОБИ ЩО ВИКОРИСТОВУЮТЬСЯ

Модулі розроблено у текстовому редакторі коду Visual Studio Code, на комп’ютері, що використовував операційну систему macOS High Sierra. Мовою програмування було обрано JavaScript та фреймворк React Native. Середовищем для тестування мобільного додатку був Genymotion Android Emulator. Мобільний додаток написаний для операційної системи Android.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Для запуску мобільного додатку на мобільному пристрої необхідно встановити apk файл на мобільний пристрій та за допомогою влаштованого сервісу завантаження програмних додатків встановити додаток. Також можна скористатися магазином мобільних додатків Google Play та скачати даний мобільний додаток звідти.

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є текст, який вводиться користувачем при створені теми дипломної роботи, редагуванні електронної пошти тощо.

Вихідними даними є списки тем, викладачів, заявок, графік виконання, а також інформаційні сторінки, наприклад, екран лабораторій.